



Volume II: Issue 6: Price £2.50

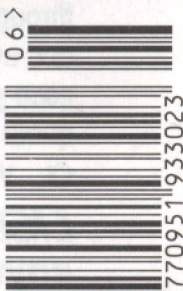
PSION
XCHANGE
free for QL

**SERIAL
MOUSE**

QL SCENE
QXL
in America

**Beginners
Machine Code**

ISSN 0951-9335



9 770951 933023



Contents

VOL 2 ISSUE 6

Editor

Helen Armstrong

Publisher

Mark Kasprowicz

Advertising Manager

Jim Peskett

Creative Director

John Stanley

Graphic Artist

Stevie Billington

Magazine Services

Linda Miller, Frances Maxwell,
Pauline Wakeling

Sinclair QL World,
Published by Arcwind Ltd.
The Blue Barn,
Tew Lane, Wootton,
Woodstock,
Oxon. OX7 1HA
Tel: 0993 811181
Fax: 0993 811481
ISSN 026806X

If you have any comments or difficulties please write to the editor and we will do our best to deal with your problem in the magazine, though we cannot guarantee individual replies.

Back issues are available from the publisher price £2.50 UK, £2.99 Europe. Overseas rates on request.

Subscriptions from: Arcwind
The Blue Barn, Tew Lane,
Wootton, Woodstock, Oxon.
OX7 1HA
UK: £23.40
Europe: £32.90
Rest of World: £40.90

Reprographic Services:
Eclipse, Brook Street,
Watlington, Oxon. OX9 5JH.
Distributed by: Seymour Press
Ltd., Windsor House, 1270
London Road, Norbury,
London, SW16 4DH

© 1993 ARCWIND LTD.
Sinclair QL World is published 12
times a year. All rights reserved.
Reproduction in whole or in part
without written permission is
strictly prohibited. We welcome
contributions. All material must be
supplied with a SAE.

While all reasonable care is taken
in compiling QL World, the
publisher and its agents assume
no responsibility in affects arising
therefrom. Opinions expressed
are those of the authors.

8 TROUBLESHOOTER

Bryan Davies looks at speed benchmarks and pointer windows.

11 QL SCENE

QXL in USA (and Birmingham) ... Italy show in September ...

12 OPEN CHANNEL

Z88 repairs ... Minerva latest ... no Abacus patch ... coded message ... musical help wanted ... the nature of Basics

15 QL SCENE

Psion Xchange freely available ... demo programs ... QReview new ... updates... QXL in Birmingham

20 VERY BASIC SUPERBASIC

Dilwyn Jones LOOPS and nests in part 3 of his Basic for Beginners.

23 THE NEW USER GUIDE - part 25

The Keyword Index from RELEASE_TASK to (please!) RETURN

27 INSTANT ACCESS

At-a-glance guide to your software and hardware suppliers.

28 REVIEW: HESSLER SER MOUSE

Bryan Davies tries the whole kit: mouse, software, cable.

31 THE HERMES INTELLIGENT IPC

Rich Mellor with a user report.

33 DIY TOOLKIT

Simon Goodwin's HPDUMP screen image printer, part 2.

37 QL SCENE

I/O from Falkenburg ... more Spectrum emulation ... Ergon updates

38 PSION SOLUTIONS

Howard Clase with two tips for getting the most out of Quill.

41 MACHINE CODE FOR BEGINNERS - PART 3

Alan Bridewell starts moving around his memory.

44 SUPERBASIC IN ACTION

Simon Goodwin trains one-colour printers to print full colour.

46 MICRO ADS

Coming Soon

Make the best of QL bulletin boards ... Easy with Easel ... DataDesign update ... Merz QVME board ... more about the QXL

Trouble Shooter

Bryan Davies looks at Eros Forenzi on speed benchmarks and Ian Turner on Pointer Windows.

The QXL card - a QL-in-a-PC - has undoubtedly aroused interest. There are few major developments on the QL scene these days, and anything new is bound to excite discussion, but something from Miracle Systems is always a special event (see the Miracle Systems advert in this month's issue). As of late May, few people had seen the card working, but the fact that it has been demonstrated suggests that it is close to being available for sale. Maybe by the time you read this, the first happy buyers will be experiencing performance that makes even the Gold Card seem slow!

There are various types of user to whom the QXL might appeal. A user with some very demanding software, which even the Gold Card cannot run fast enough, could find it worthwhile to buy both the QXL and a cheap PC; simply to get the fastest "QL" available. There are some very cheap PCs on the market, but it is desirable to look long and hard before parting with cash, as the cheap deals may not be good ones. The cheaper systems are generally old and lack some of the facilities which may be required; for example, they are unlikely to have high-density floppy drives and may not have 3.5-inch drives at all, the hard disk drives (if any) will be low-capacity, the display graphics may be only mono or CGA colour, the keyboard may be the old XT style not the current AT style. The QXL card can be used in either an 8- or a 16-bit expansion slot but requires EGA-level graphics capability.

If you are considering getting an old Amstrad PC1640, make

sure it is one with EGA display capability; that is, an ECD model, not the CD or MD types. The display is marked with the designation. The computer unit itself is marked with the designation for the drive types, and you need HD for the hard disk model.

Software87

An early report on the combination of Text87 with Line Design says that the results are good. Graphics and text can be printed together. The clip art and founts supplied with Line Draw are said to be good, and the founts can be scaled to a wide range of sizes. Sample prints certainly suggest the package is a good one; the graphic images are well-drawn, and print out well. Text founts are somewhat variable, but the best of them are good. Very much to the point, when you get into the DTP area, the necessary operations for producing mixed text-and-graphics documents are not too complicated.

Software87 are also working on drivers for the Hewlett-Packard LaserJet 3 and 4, and on a utility program to allow character translation strings in printer-drivers to be altered without recourse to an assembler program. The aim of the latter is to permit the use of "special" characters such as box-drawing ones and mathematical symbols.

Bits and pieces

As of mid-May, the version of the Minerva rom being shipped was 1.97. The IPC 8049 chip in the QL is not without its faults and it is worth considering

replacing it by the Hermes chip (from TF Services). Stated merits of the Hermes are the setting of different Baud rates for SER1 and SER2 (both ports have to run at the same rate on the standard QL), curing of "keyboard bounce" with add-on keyboards (eg Schoen, Keyboard Products), improved handshaking with serial input, and maintenance of the set pitch when sound functions (eg fuzzy and random) are used.

Details on Arcplus, the enhanced PC version of Archive, have been supplied by Transform Ltd. This program is not cheap and there would have to be a serious, business requirement for it to justify the cost, and the work of porting-across existing QL Archive Procedures. However, Archive is still a good program and there are no doubt plenty of users who wish to stick with it, but perhaps need the extra features that Arcplus can provide.

Speed test

Eros Forenzi, one of the prime movers behind the QItaly user group, has supplied a disk containing a speed-testing routine, the QL Speed Index (QSI), for the QL and its derivatives. Unusually, the routine runs under Archive. The aim has been to create a benchmark test that can be applied to any variation on the QL theme, and to distribute it widely enough to make it a standard test, enabling sensible comparisons to be made in future. This means the routine is in the public domain, to be used and copied freely. Eros compares it to the Landmark test used on PCs but, hopefully, his test is more meaningful than that one.

It was not possible to reproduce the Archive routine

code here in the time available, but please write to me care of QL World, enclosing a 3.5-inch disk and a stamped, addressed return envelope if you want a copy.

As Eros says, we in the QL world have not had to bother about benchmark tests as much as users in the PC world do, but we are getting to the same situation now. Benchmark test figures can easily become very misleading if they are not applied with great care. Basic QL, QL with old disk/memory interfaces, Thor I, Thor 20, Thor XVI, QL with Trump Card ('slow' and fast), Atari with (hardware) QL emulator, QL with Gold Card (three varieties, at least), Amiga with (software) QL emulator, and now the PC with QXL - there are plenty of configurations to compare. Eros has not yet got test figures for several of these configurations, and he asks that users running his routine advise him of the results (see INFORMATION for his address), if their QL configuration is different from the ones for which he lists figures. He suggests results be tabulated like this:

My Qdos system is:
Computer: eg Atari
Processor: eg 68030
Clock: eg 25 MHz
Other details: eg cache ram disabled
Archive version eg 2.30

Test results are:
Elapsed time: eg 900 secs (as read from screen)
QSI QL index
QSI Gold Card Index

QSI Absolute speed

The routine gives three results for the speed of the system being tested - relative to a standard 128 KB QL, to a QL with 16 MHz Gold Card, and in

	TIME (SECS.)	QL INDEX	GC INDEX	ABS. SPEED (GC = 100)
Standard 128 KB QL	800	1.00	0.15	15.38
QL + 512 kB Expanderam	588	1.36	0.21	20.92
QL + Trump Card 2	467	1.71	0.26	26.34
QL + Gold Card 16 MHz	123	6.50	1.00	100.00
QL + Gold Card 24 MHz	83	9.64	1.48	148.19

Left: table in reference to the "Speed Tests" article.
Below: listing in reference to "Pointer Windows" article

absolute terms (where the 16 MHz Gold Card system = 100). The configurations Eros has data for are shown in table one.

The source (inspiration?) for the routine was a commercial task Eros had to tackle. This was a requirement to convert (money) numbers into letters, for use in Archive. For example, changing 100 to ONE HUNDRED. Anyone more impressed by the original Italian might prefer 1002110 coming out as UNMILIONEDUEMILACENTODIECI. The test routine performs 100 such conversions and they are printed at the same position on the screen; the characters are printed in black ink and you cannot see anything - as Eros says, they would not make much sense to non-Italians, anyway! There is no apparent problem with testing machines fitted with different roms (JS, JM, Minerva). The only restrictions on running the test are that the standard form of Archive (not ARCHRTM or ARCHDEV) should be used, neither TK2 (Toolkit 2) nor any other toolkit extensions or accelerator functions should be switched on, and no other program should be multi-tasking with the benchmark. You can multi-task programs and use speed-up software such as Lightning if you wish, but that will make the results invalid for comparison with others taken with "clean" systems.

Pointer windows

Ian Turner wrote in some months ago, on the subject of the Pointer Environment. In trying to understand the workings of the PE, he went through the tutorials provided on the subject, but apparently did not get too far. He did, however, learn something about re-sizing windows, and he sent in an

assembly-code program for doing that job (listing one). Some minor changes have been made to fit the code onto the page; it is public domain and may by now have been put into the Quanta library.

Two German QL enthusiasts gave us disks at the Eindhoven show, and they may be wondering what happened to them. They are sat here in front of me at the moment, but there has not been time to digest what is on them and make comment in this issue; all being well, they will have been looked at more thoroughly in time for the next issue.

Switches have been in use on printers for years - I have one printer switched between two computers, and separately connected to a third - but what about switchable mice? That may sound a daft question, but mice do create a space problem, and my workbench presently has three active mice plus two mouse pads sat on it. Apart from taking up a lot of space, the varmints cause confusion - you sometimes curse because the mouse you are pushing around is not interested in the

Listing One

```
* RESIDENT PROCEDURE TO SET WINDOWS #0, #1 AND #2
* FOR THE POINTER ENVIRONMENT
*
* Program Name : WPTR
* Written by   : Ian Turner
* Date Written : 16 Jan 93
*
* Window Parameters:
*
* #0 - 512,52,0,204 - Border 1,224 - Ink 4, Paper 0
* #1 - 256,182,256,16 - Border 1,255 - Ink 7, Paper 2
* #2 - 256,182,0,16 - Border 1,255 - Ink 7, Paper 0
*
BLACK EQU 0
RED EQU 2
GREEN EQU 4
WHITE EQU 7
*
WINDOW_0 EQU 0
WINDOW_1 EQU $10001
WINDOW_2 EQU $20002
*
WINDOWS LEA.L DEFINE,A1
MOVE.W $110,A2
JSR (A2)
LEA.L MODEVAL,A2
MOVE.B #0,(A2)
RTS
*
DEFINE DC.W 1
DC.W WPTR-
DC.B 4,'WPTR'
DC.W 0,0,0
*
WPTR MOVE.W $112,A2
JSR (A2)
TST.L D0
BNE.S WPTR_EXIT
CMP.W #1,D0
BGT.S BAD_PARAM
BNE.S PAPER_0
MOVE.W #2,$58(A6)
ADDQ.L #8,D1
ANDL.W MODE_8
BNE.L LEA.L MODEVAL,A2
LEA.L #0,(A2)
MOVE.B PAPER_0
BRA.S PAPER_0
*
BAD_PARAM MOVEQ #-16,D0
WPTR_EXIT RTS
*
MODE_8 LEA.L MODEVAL,A2
MOVE.B #8,(A2)
*
SET_WINDOW #0
PAPER_0 MOVE.L #WINDOW_0,A0
MOVEQ #BLACK,D1
LEA.L PAPER,A2
JSR (A2)
*
INK_0 MOVE.L #WINDOW_0,A0
MOVEQ #GREEN,D1
LEA.L INK,A2
JSR (A2)
*
SIZE_0 MOVE.L #WINDOW_0,A0
LEA.L DEFINE_0,A1
MOVEQ #500,D0
MOVE.B #0,D1
MOVE.W #0,D2
MOVE.W $FFFF,D3
TRAP #3
*
BORDER_0 MOVE.L #WINDOW_0,A0
MOVE.B #224,D1
LEA.L BORDER,A2
JSR (A2)
*
SET_WINDOW #1
PAPER_1 MOVE.L #WINDOW_1,A0
MOVEQ #RED,D1
LEA.L PAPER,A2
JSR (A2)
*
INK_1 MOVE.L #WINDOW_1,A0
MOVEQ #WHITE,D1
LEA.L INK,A2
JSR (A2)
*
SIZE_1 MOVE.L #WINDOW_1,A0
LEA.L DEFINE_1,A1
MOVEQ #500,D0
```

INFORMATION

Arcplus; Transform Ltd., 7c Station Approach, Hayes, Kent
BR2 7EQ Tel. 081 462 4666 Fax 081 462 3971
QSI benchmark; Eros Forenzi, Via Valeriana 44, 3010
Berbenno (SO), Italy. Tel. +39-342-492323 (after 1900 hrs GMT)
Minerva and Hermes chips; TF Services, 12 Bouverie Place
London W2 1RB, Tel. 071 724 9053 Fax or scrolling modem
071 706 2379

MOVE.B	#0,D1	* BORDER COLOUR
MOVE.W	#0,D2	* BORDER WIDTH
MOVE.W	#FFFF,D3	* INFINITE TIMEOUT
TRAP	#3	
BORDER_1	MOVE.L	#WINDOW_1,A0
	MOVE.B	#255,D1
	LEA.L	BORDER,A2
	JSR	(A2)
SET WINDOW #2		
PAPER_2	MOVE.L	#WINDOW_2,A0
	MOVEQ	#BLACK,D1
	LEA.L	PAPER,A2
	JSR	(A2)
INK_2	MOVE.L	#WINDOW_2,A0
	MOVEQ	#WHITE,D1
	LEA.L	INK,A2
	JSR	(A2)
SIZE_2	MOVE.L	#WINDOW_2,A0
PIXELS		
	LEA.L	DEFINE_2,A1
	MOVEQ	#00,D0
	MOVE.B	#0,D1
	MOVE.W	#0,D2
	MOVE.W	#FFFF,D3
	TRAP	#3
BORDER_2	MOVE.L	#WINDOW_2,A0
	MOVE.B	#255,D1
	LEA.L	BORDER,A2
	JSR	(A2)
SET MODE		
MODE_SET	MOVE.W	#10,D0
	LEA.L	MODEVAL,A2
	MOVE.B	(A2),D1
	MOVE.B	#-1,D2
	TRAP	#1
	LEA.L	MODEVAL,A2
	MOVE.B	#0,(A2)
RTS		
WINDOW SUBROUTINES		
PAPER	MOVEQ	#27,D0
	MOVE.W	#FFFF,D3
	TRAP	#3
	MOVEQ	#28,D0
PAPER)		
	TRAP	#3
	RTS	
ING		
INK	MOVEQ	#29,D0
	MOVE.W	#FFFF,D3
	TRAP	#3
	RTS	
BORDER	MOVEQ	#0C,D0
	MOVE.W	#1,D2
	MOVE.W	#FFFF,D3
	TRAP	#3
	RTS	
WINDOW SIZE DEFINITION BLOCKS		
DS.W	0	
DEFINE_0	DC.W	512
	DC.W	52
	DC.W	0
	DC.W	204
DEFINE_1	DC.W	256
	DC.W	182
	DC.W	256
	DC.W	16
DEFINE_3	DC.W	256
	DC.W	182
	DC.W	0
	DC.W	16
MODEVAL	DS.B	1
END		

screen you are looking at. An infra-red mouse (see review of the SERMouse mouse driver) does relieve some of the cable clutter, but one has to face the fact that computer systems consume large amounts of space, and that space is needed for other things, such as the paperwork the systems generate. Maybe one infra-red mouse and three infra-red receiver units would be an answer for me, but what happens if more than one computer reacts to the mouse activities? Amazing how computers provide endless occupation, without driving one to terminal distraction or boredom.

TF SERVICES

MINERVA

The ULTIMATE operating system upgrade

MKII MINERVA with battery for 256 bytes ram. CRASHPROOF clock & Philips PC bus for interfacing. Can autoboot from battery backed ram. Quick start-up.

Other features common to MKI/MKII
DEBUGGED operating system/ autoboot on reset/ Multiple Basic/ faster scheduler-graphics (10% of lighting)-string handling/ WHEN ERROR/2nd screen/ TRACE/ foreign keyboard drivers/ 'warm' fast reset/ Auto reboot after power failure. V1.97 now with built in Multibasic and split OUTPUT baud rates with Hermes.

1st upgrade: free. Otherwise £3 (+ £5 for manual if reqd) (send sac, rom & NEW disk/3 mds)

MK1 to MKII upgrade - £30

MK1...£40

MKII...£65

BOTH VERSIONS GOLD CARD COMPATIBLE

HERMES

A replacement QL co-processor for the QLs awful IPC 8049

- Do you get keyboard bounce?
- Do you find fast serial input unreliable?
- Do you want to connect a modem at 19200bps

If you can say YES to any of these, then you need HERMES

- 19200bps RELIABLE serial input - NO QCONNECT.
- Independent input baud rates - use serial mouse & print
- Stops keyboard bounce (unwanted repeat chrs)
- Improves 'fuzzy' and 'random' sound
- Provides extra input/output lines
- Key click

Fitting is simple. Remove the QL top (8 screws) & replace the chip marked 8049 or 8749 next to mdv 1.

£25 including manual/software

QL SPARES

Faulty QL board (no plug-in chips).....£9

Keyboard membrane.....£9	Circuit diagrams.....£2
68008 cpu.....£8	1377 PAL.....£3
JM ROM.....£10	Power supply.....£12
3302 ULA.....£10	8301 ULA.....£10
8049 IPC.....£8	MDV ULA.....£12

Other components/sockets etc) please phone

QL REPAIRS

Fixed price for unmodified QLs, excluding microdrives.
QLs tested with Thorn-EMI rig and ROM software

£27 including 6 month guarantee

All prices include post & packing (UK only). Payment by Mastercard/VISA/Bancontact/cheque/postal order/PO Giro transfer (S8 267 3909). MAIL ORDER ONLY - no callers without ringing first. Ring for overseas prices.

VISA

12 Bouverie Place, LONDON W2 1RB

Tel: 071-724 9053

Fax: 071-706 2379

MasterCard

Telephone: 0753-888866

Fax 0753-887149

(EEC)

W.N. Richardson & Co.

SINCLAIR QL PRICE LIST MARCH '93

18-21, Minbourne House,
Chalfont St Peter, SL9 8UE

All Prices Include 17.5% VAT

QL Computers

NEW MONITOR £50 CHEAPER

JS COMPLETE

QL Computer, PSU, T.V. Lead, PSION V2.35 Software (Word Processor, Spreadsheet, Database & Business Graphics). Handbook for QL Programs, & Superbasic. Free one year membership to QUANTA, the independent QL User Group with over 400 free library programs, Newsletters & HELP. 6 MONTHS WARRANTY. WITH JM ROM

JS £120

PART EXCHANGE £30 OFF ABOVE. Send in old QL (Unit only, any condition) - JS ROM £90 JM £70.00

BACKUP QLs QL & PSU only. JS £80 JME65 (Part exchange allowance £15 if required)

Accessories

* NOTE EXTERNAL (SER2) 3 BUTTON MOUSE AND SOFTWARE WITH EXTRA FUNCTIONS. NOW HERMES COMPATIBLE *

PC KEYBOARD INTERFACE, INTERNAL FITTING, FOR FITTING 102 KEY KEYBOARD	£25.00
PC KEYBOARD, UK version, 102 key (AT)	£30.00
PC KEYBOARD INTERFACE AND KEYBOARD (INCLUDES FREE JOYSTICK OR PSU)	£95.00
CASE AND LEAD FOR EXTERNAL FITTING of Keyboard Interface	£14.00
JOYSTICK with QL lead (no interface required)	£10.00
* QL MOUSE, 3 Button, Software controlled, externally mounted, simply fits in SER2	£45.00
CORDLESS MOUSE	£55.00

Disk Drives

THE UNIVERSAL DRIVES ARE ALSO SUITABLE FOR ACORN BBC, ATARI ST, AMIGA, SPECTRUMS, IBM, AMSTRAD AND OTHER PC COMPATIBLES

Universal 3.5" 1Mb disk drive, cased with PSU and free QL lead	£70.00	2 Units for dual drive	£120.00
Universal 3.5" 2Mb disk drive, as above but 2Mb capacity	£90.00	2 Units for dual drive	£170.00
3.5" 1Mb Uncased disk drive			£25.00
3.5" 2Mb Uncased disk drive			£40.00
Lead for uncased disk drives, or other universal micros	£10.00	POWER SUPPLY FOR UNCASSED DRIVES	£6.00
		METAL CASES FOR UNCASSED DRIVES	£6.00

Parallel Printers

NEW RANGE

SAMSUNG SP093 80 Col, 300 cps, 50 NLO, 3K Print Buffer, Centronics, Tractor Feed/Sheet Feed, Paper Park OLIVETTI COLOUR PRINTER, 24 pin, 200 cps, 50 NLO, 40K Buffer, Epson LQ2550/IBM Compatible

CENTRONICS PARALLEL Printer Interface for above printers

QL LEAD FOR SERIAL PRINTERS £10

CANNON BJ10 EX

£149

OK FOR PC AND QL SELF SENSING AND MANY EXT'L CONTROLS.

* MONO OPTION* WITH 9" GREEN SCREEN

8 prog cards for formatting in 2 wallets

£15.00

QL power Supply Unit

£10.00

TV or Network leads

£3.00

IC's ZX 8301 £6.00 ZX 8302 £3.00

MC 68008 £12.00

QL SERVICE MANUALS & CIRCUIT

£25.00

S.A.E. FOR FURTHER DETAILS

Payment terms: CWO, Access, VISA at cost

Cheques - allow 10 days

Product subject to availability, E&OE

TEL: 0753 888866

FAX: 0753 887149

Delivery: Carriage - £9

Postage - £5

JUNE
10

SINCLAIR QL WORLD

Psion open door to **QL Xchange**

Following ground-breaking enquiries by Mike Bedford-White and Simon Goodwin of the West Midlands Quanta group, and a communication from Helmuth Stuver, formerly of Thor International and Dansoft, QL World contacted Psion to check the status of all versions of Psion Xchange for the QL known to us.

Psion Xchange is Psion's name for the suite of four programs - Quill, Abacus, Archive and Easel - that were bundled with the new QL from the start of its career. Various upgrades of these programs have appeared, notably Thor Xchange, a partially-rewritten version produced by Dansoft for the Thor.

Further upgrades and adaptations of the programs have been produced privately by programmers who want to see them widely available to QL users. QL World's enquiry was about the copyright and distribution status of the four programs. Psion's replied:

"Thank you for your letter regarding Psion Xchange. I am happy to clarify the position. We are quite happy for the four software programs, Abacus, Archive, Easel and Quill for the Sinclair QL or derivative machines to be freely distributed for use on the QL and derivative machines. We retain copyright at all times.

"The re-write made for the Thor computers by Dansoft was done under licence from us, and we retain copyright to part of the code. As described in the point above, we are happy for this to be distributed without charge while retaining copyright.

"The product PC-Four for the IBM-PC or machines running MS-DOS emulation is not included in the above, and remains a product we sell. We would be most unhappy to find this product in the public domain.

"In summary, we are happy for QL specific (or derivative products) software to be distributed freely to users while retaining our rights. We are not prepared to allow IBM PC (or derivative) versions to be distributed freely.

"We trust this makes it perfectly clear, and I trust that this will not prevent the QL community from continuing to enjoy their Psion software.

"Peter Norman, Group Commercial Director, Psion PLC, London NW8."

This letter confirms Psion's letter to West Midlands Quanta, and makes it clear that they are happy for the whole QL public to enjoy free distribution of the Xchange programs for the QL. It also clarifies that Psion retain all their copyrights in the programs, which can be distributed and adapted, but not cannibalised for commercial programming. The status of Psion Xchange programs for the QL is somewhat like Shareware, but without registration fees, documentation or support.

Psion also make the point that Xchange for the PC (PC-Four) is a commercial product not free for copying by users.

Helmuth Stuver's letter to QL World earlier this year donated the programs bundled with the Thor in which he had an interest to the public domain. All code belonging to Psion (including that incorporated in Thor Xchange) is governed by Psion's terms, but the effect to the QL user is the same: the programs can be freely distributed.

Ex-Dansoft workers Gunther Strube and Erling Jacobsen have modified Thor Xchange 3.90 to run on the QL and compatibles. This version can be obtained from them by sending a formatted 720K QL disk and sufficient International Reply Coupons for return post to Gunther Strube, GL Kongevej 37, 2nd, DK-1610 Copenhagen V, Denmark.

Thor International also supplied Psion Chess in a disk version for the Thor. This program has been adapted to execute as a job (CALLd on the QL, and not multitasking). A copy is available to users sending a disk and reply coupons, plus their original Psion Chess microcassette as proof of purchase. Psion Chess is not covered in Psion's release terms for Xchange.

Of the various versions of Xchange for the QL in existence, most should be available soon from public domain suppliers for their usual copying charges, or from a friend.

Established users claim that the Xchange upgrades for the QL offer considerable improvements on versions originally bundled with the QL.

Demos hit town

The latest additions to SJPD's lists are Xchange 3.90 from Psion, two disks of C programs from Richard Kettlewell, comprising YACC (a compiler), MAKE (a new version), C Toolkit, Flex and utilities; a David Walker Demo Disk with versions of Textidy, MultiDiscover and Discover, and a shareware version of FLPClone disk copier; a demo disk of Di-Ren's Fleet Tactical Command 2 (QL and PC versions available); Speculator, the new Spectrum emulator in English; QL-REXX, a REXX interpreter for the QL, and DME, a Pointer Environment (not supplied) text editor ported from the Amiga.

SJPD's printed catalogue is available for four first-class stamps. For the up to the minute information, their disk catalogue is available free to anyone sending a formatted disk, return postage and an address label. This disk also contains some free sample software.

SJPD's new arrivals reflect how keen many software writers are to produce demo versions of their favourite programs at a low cost, so that QL owners can see them in operation and try some of the functions before pledging large sums of money.

Contact: SJPD, 36 Eldwick St., Burnley, Lancs. BB10 3DZ. Tel. 0282 451854.

QLScene

New - QReview

We have just heard that Bruce Nicholls, publisher of QL Leisure Reivew and QL Technical Review (formerly published by CGH) is closing those two titles and publishing a new title, QReview. Existing subscribers will continue to receive the new title, and back issues of the previous titles will continue to be available from Bruce at Quo Vadis Design, and also from Dilwyn Jones.

More detail next month. Enquiries to **Bruce Nicholls at Quo Vadis Design, 57 Shaftesbury Road, Romford, Essex RM1 2QJ. Tel. 0708 755759.**

Open Channel

Open Channel is where you have the opportunity to voice your opinions in Sinclair QL World. Whether you want to ask for help with a technical problem, provide somebody with an answer, or just sound off about something which bothers you. Write to: Open Channel, The Blue Barn, Tew Lane, Wootton, Woodstock, Oxon. OX7 1HA.

Dead 88

I recently bought a second hand Z88 notepad computer. Unfortunately, it has already died on me, and the seller is nowhere to be found (caveat emptor...). As so many QL owners have one of these, I would be grateful if anyone could let me know where I could have mine repaired and at what cost. I have written to the Z88 users group but without success. It's possible that I had an incorrect address.

Also, does anybody have or know of a Quill printer driver for the Citizen Swift 24? (I believe it can emulate the Epson LQ500 among others.) I have never heard of such a driver, but I live in hope.

Jean-Yves Rouffiac
Gospel Oak
London

In 1990 the Z88 handbook listed the following in the London area as Z88 registered dealers: Radio Shack in NW6, Hardware Trade Tech in NW10, KK Stationers in W2 and Micro Anvika, PC Consultants, Gultronics, Megateck Computers and G&B Computers in W1. Addresses and phone numbers are constantly shifting, so it's over to you from there. Otherwise, it's a matter of finding a reputable computer dealer in your area and getting a quotation for the repair.

But your best alternative is to contact Cambridge Computer. Their office number is now 0294 222100, but they have a repair depot in Cambridge at EFS Ltd., Unit 4, Chesterton Mill, French's

Road, Cambridge CB4 3NP. Tel. 0223 353355, and ask for Amos Fulcher.

Caveat emptor! You can't normally claim recompense from the seller of a second-hand computer if it was sold as seen in good faith.

Catch a Mouse

It was nice to meet you in Eindhoven. As pointer software gets more attention in the UK, also the interest of connecting a mouse to the QL seems to be growing. My SERmouse driver for a serial PC mouse seems to be the only available mouse solution of its kind for the QL, and I think it would be interesting for your readers to find a review in *QL World*.

Albin Hessler
Aichtal
Germany

It's in this issue. The Qimi house interface, and the QLMUG mouse interface and still available if you know where to look for them - try Quanta. Albin's mouse software development has the Pointer Environment very much in mind. See page 28.

Minerva-wise

I was disturbed to see Bryan Davies' speculations about Minerva in the April Troubleshooter. Firstly, a computer that doesn't develop will die all the sooner. I find it staggering that he expects the operating system to be archived

in 1986. There are some 1200 Minerva users out there (based on my sales records). It is not realistic for him to dismiss Minerva as "tinkering".

Also, he reported problems of using a "released" version not working with *Professional Publisher* or the *Pointer Environment*. In fact, this was a beta test version! I have discussed this with him. I also asked him what the problem was with the *Pointer Environment*, and he said he didn't know. The only problem we had was something Laurence (Reeves) was trying to do with Multibasics. When we found it didn't work on beta test we removed it.

The released version 1.97 doesn't have problems with either the *Pointer Environment* or *ProPub*, to our knowledge. The current problems are now very few, when the suggestions made in our documentation are followed. I would be grateful if you could correct the false impression given.

Tony Firshman
TF Services
12 Bouverie Place
London W2 1RB

Glad to. When Minerva first appeared, based as far as possible on the most advanced Sinclair QL rom developments, there was a bit of friction about differences with the rom which software writers had been working with. One party of supporters argued that software writers had to follow the most widely-used standards and that new developments should take this first into consideration. The other party argued that the Minerva rom was of a purer,

older lineage and did things properly, and that software developers should be adapting to that, not the other way round.

Since the dust died down, quiet and steady work on both sides has meant that there are now very few incompatibilities. The ideological gulf can be argued both ways, but in reality people have adapted to get the best of both worlds.

The East Anglia Quanta Group also market a neat rom-switcher for those users who want to use two different roms. The contact is Chris Howard at 13 Oak Grove, Horsford, Norwich, Norfolk NR10 3DR, tel. 0603 891183.

People close to the beta-test circuit often seem to be uncertain which is the current release of Minerva, but Tony always knows what the latest situation is and will always give the best advice. Speak to his answering machine on 071 724 9053, or fax/scrolling modem 071 706 2379.

Not a patch

E B Palmer (*Open Channel*, April 1993) whether there is any way the abacus layout can be expanded beyond 255 rows and/or 64 columns.

Having seen this question in your columns before, may I lay the ghost once and for all? I looked at this problem a few years back, as I too was annoyed by the limitation imposed by 256 rows and, at the time, no other QL spreadsheet had seen the light of day (there is now *QSpread*). I have since chucked my notes away, but the following thoughts from memory may clarify the

position.

To put it bluntly, there is no possibility of a patch to Abacus to increase the number of cells. In the first instance, all the Psion suite of programs were originally coded in high-level code, then cross-compiled for the QL; later versions of the Psion suite were, I believe, partly hand coded in machine code to optimise them, but the resultant code is, nevertheless, very hard to understand and hence patch.

Worse still, the 256 row limitation is imposed in the machine code by the use of word length pointers to cells (8 bits (one word) allows the representation of decimal 0 to 255, no more). The 64-columns limitation is similarly imposed (6 bits gives decimal 0 to 63, the other two bits are used as formatting flags). To increase the size of the spreadsheet therefore, all instructions which use the cell pointers would have to be changed from word-length to double-word-length (16 bits). If one imagines how much of the program is devoted to manipulating the contents of cells (almost all of it!), it can be seen that the change required is not a patch, it is a total rewrite! Without the original annotated source code and developers; documentation, this would be likely to take longer, and be more expensive, than writing a new spreadsheet program from scratch. I am afraid Mr. Palmer will need to look elsewhere, possibly, as Howard Clase recommended, at *QSpread*, for his solution.

Malcolm Bacchus
London SE14

QSpread has an increasingly enthusiastic following among QL users as word gets around.

Connections

Regarding my earlier letter (March 1993) about the joys of Qram and similar programs, since I wrote that letter Mr. Dolezal has been in touch with us and sent us a cheque for £30 as a refund for the Q-Top which I returned because I do not have Toolkit 2.

On to other things. I have a Tandata communications unit with which I used to use Prestel, and to communicate both to and from other numbers until I gave up Prestel. I would like to see more in the magazine about communications over the telephone network, because I have no contacts since giving up Prestel. I have tried the

numbers of so-called bulletin boards but with no connection to date.

Also, I have two Psion Organiser XPs with 32KB which I use even more than I use my QL, one with the wordprocessor and one with the spreadsheet. I have the Textbase database program for both of the XPs, and can transfer from the XP to the QL Archive. I would use the QL even more if I could transfer script and numbers to it from the XP direct instead of through Textbase and Archive.

Is there sufficient interest in such subjects to have more information about them in the magazine?

Lawrence Carpenter
Guernsey
Channel Islands

We've got something on bulletin boards coming up. Try talking to Tony Firshman at TF Services (via his answering machine) and Bob Weekes at Pointer Products (0258 455117 voice and fax), who both run bulletin boards. I'm always interested in hearing from people who are piping data between the QL and other computers, to learn their methods. We'll look at putting together an article on it if enough people are interested.

Enigma

Why are all your contributors so serious? Is there no-one out there who uses the QL to have fun? And I don't mean just playing computer games. I notice that you have just started a new series of articles for the absolute beginner. Can there still be people who look glassy-eyed at the box and wonder what to do with it?

Take my case, for example. I have had a QL for about six years, working up from a ZX81 and a Spectrum. I can't understand multi-tasking and don't see why I should want it anyway. Two-thirds of the instructions in my Toolkit 2 are a mystery to me, and a fair bit of the manual besides. If I don't use an instruction for more than a week, I have to learn it all over again. But I get a great deal of pleasure in writing a program in SuperBasic to do something probably quite useless, and trying and eventually succeeding in making it work. Am I the only person among your readership who operates on this level?

For example, I have written a program to simulate the

operation of the German wartime coding machine Enigma. How about publishing a small message in code and challenging all the experts to decode it and to explain how they did it? To give them a chance you would need to publish the program as well.

George Phelps
St. Leonards
East Sussex

That's the best sales pitch I've heard for some time.

When in Rome

Regarding your article on the QL International Meeting, a small correction: the photo showing the QItaly stand was really the Ergon stand! I am the first from the left sitting, and then there is the 'artificial intelligence' guru Marco Ternelli, who was showing ZM/hT to Roberto Orlandi of QItaly Club.

Keep up the good work on QL World.

Davide Santachiara
Reggio Emilia
Italy

You're quite right! QItaly had a banner slung along the pillars behind the stand; when they weren't circulating, Eros and Roberto were 'hanging out' with the Ergon bunch - doubtless

Editor's notebook

Probably the most interesting news this month - and it's not news, of course, to those who have been working on it - is that Psion have definitely declared that they are happy for versions of Psion Xchange (the Psion Quartet) for the QL and its compatibles to be circulated freely by all QL users.

Psion are not the first copyright owner to take the view that, once a piece of their software bundled under licence is no longer being actively marketed, upgraded and supported, then, provided there is no trespass on the owner's activities, users of the machines concerned are free to work on and adapt that software.

This does not amount to Public Domain rights, but it does mean that QL (and Thor) Abacus, Archive, Quill and Easel can be freely upgraded, copied and shared. More information in QL Scene.

We found Mike Lloyd. He is a very busy man at present, but you may be sure that we are contributing to his busy-ness!

planning the joint QL Italy Meeting this coming September (see QL Scene).

Cheers!

Thank you for QL World - reassuring, inspiring, and always the prospect of even better things to come. You are all greatly appreciated.

Malcolm MacLeod
Aberdeen

Thank you, Malcolm. Renewals with the personal touch!

Late letters

(Slightly condensed): why do some letters get printed a long time after they are sent?

Alan Ingreay
Peterborough

A few letters get hived off so that I can ask around for answers. If I don't get an answer, I run the letter in Open Channel, say, three or four production-cycles later (or whenever the letter re-emerges from the system), in the hope that it will be seen by someone who has the answer. I forward several "answer" letters a month, and print some of them, so it works pretty well. QL World is in the business of sharing problems

and ideas, so appeals in a publishable form are the best kind.

Our "lead time" for urgent things is down to three or four weeks now the best it's ever been.

Take notes

Help! I urgently need a method of writing musical manuscripts on my QL. I would prefer to be able to connect a keyboard via a Midi interface and have the computer print the music as I play, or at least compile the music in notation form to print later.

I cannot see any way to purchase a Midi interface for my QL. Is there a company that sells these devices? I don't want to have to go to the great expense of converting to a PC to accomplish this.

J G Wilkinson
Southampton

Yes! Quanta themselves market the former *Miracle Midi* interface. As for software, there was, two or three years ago, a decent notation program for the QL, which was reviewed for us by Eleanor Patrick, who is a Quanta member in the Leeds area. I don't think it fed notation straight to memory, though. Not many low-cost programs do. Try Quanta chairman Bill Newell at 213 Manor Road, Benfleet, Essex SS7 4JD and see if he can assist.

Basics

I spotted a couple of queries in *Open Channel* earlier in the year which I thought I might be able to answer. First, ZX Basics: the Spectrum Basic is actually closer to ZX81 Basic than the ZX81 version is to the ZX80, although there is still a large amount of extra code. There is actually a piece of code in the Spectrum rom that is not used, but is a leftover from the ZX81; presumably someone forgot to remove it!

SuperBasic being "tagged onto Qdos" doesn't necessarily mean it was designed for another CPU. The original intention - according to what I've read - was for GST to write the QL's operating system, so perhaps SuperBasic was originally designed for that.

On the subject of GST, their assembler and compiler are

now being sold to members by Quanta. The assembler has been updated a bit, and I can highly recommend it. As for the C compiler, C68 is a better bet - free apart from disk copying charges from Public Domain dealers - unless you're short of memory.

Richard Kettlewell
Bournemouth
or Cambridge
(rjk1002@uk.ac.cam.phx)

Antique disks

A while ago a correspondent reported trouble reading disks which had been written to some years previously. This has happened to me, too, and the trouble was not worn drives, as these had been laid up as long as the disks. The disks were all master copies from softwarehouses, and unbranded. Other disks stored in the same disk box (branded) were all OK. The moral seems obvious.

A particular degree of magnetism in a substance is not totally fixed and stable over time. A lower quality disk will be magnetised less in the first place and so liable to suspect reading first. Fortunately, due to their age, my disks were written to on DD drives, and could mostly be read on an ED drive which has more sensitive read circuitry to cope with the higher recording density.

The one disk I couldn't recover this way was from Sector Software. Joy of joys, they responded to the answerphone and offered to rewrite the disk for the cost of the return postage, but the suspect disk and several follow-up pleading letters seem to have disappeared into a black hole somewhere in Lancashire. Of course, I didn't have a backup. Sad.

Please, please, please don't give up on Mike Lloyd's *Keyword Index*, which has been absent for the last few months. I should be happy to pay photocopy fees to get mine, if I must. There are quite a few alternative (if elderly) publications on SuperBasic, but not on the Toolkits. I would be grateful to hear what is happening to this.

In general, things have improved post-Maxwell, apart from the occasional writer who assumes that readers know all about widgets and just need the SP on this particular implementation. What about

some editorial in a sidebar to deal with this problem? I think that the QL Lib manual is a model of clarity others should aspire to.

The quality of software seems much improved over the last few years, even if there is less of it. That does mean we may have to rely on our own programming a bit more. Hopefully this will influence your output!

Ivan Hall
Lapford
Devon

Several interesting points here. The elasticity of the time domain is one of them: Mike's New User Guide has been absent for two months, the latter one because he misread his copy date! Mike is setting up his own computer business in his spare time, and this is taking a great deal of work and brain-power. I trust he will note how he is appreciated. Often writers don't get any feedback unless there is an error!

I spend a good deal of time trying to get authors to hit the right level in articles, especially the technical ones, but it isn't always practical to take everything back to basics. If you are following an article where something isn't clear to you - drop us a line, and we'll get the author to expand on it.

Good Spell

I'm pleased to see Fred Toussi taking a high profile and accepting the praise and comments on his tremendous Text87 Plus4, and also to see his replies in *Open Channel*, as well as to have received personal communications.

The Spellchecker on his wordprocessor does more for me than any other I have tried, and I wonder at those who are content with a simpler approach, or dismiss the notion that it is necessary at all!

I am not looking for a device which corrects my syntax and semantic errors, but there is a long list of words that I commonly use - medical psychological terms, my own notations, etc., as well as common typing errors - that it is welcome to have picked out. In fact, my intention is to praise the Toussi Spellchecker in favour of my "work area" program simply because it does not come up with "guessing game" varieties of what has been typed in.

My next obvious move is to install a Gold Card with a pair of

ED drives. I can't wait! Freddy Vachha put on a tremendous demonstration of the speed possible running *Conqueror* Gold SE, which I have, although regrettably without the DR-DOS to make it usable. It has joined some other things on the shelf to await finance and time, but holds a prospect for the future.

In the meantime, before Freddy points it out (!) I have taken the opportunity to upgrade my copy of *Perfection* again, although I still prefer the text87 spellcheck side of things at the present date. I was anxious, but alas unable to take advantage of DP's "special show discounts", so this is as far as I can stretch just now!

L Ross Bayne
Reading

Open Channel

Italy in September

The 5th Italian QL Meeting - organised by Ergon Development and the QItaly club - will again be held in Reggio Emilia on Sunday 26 September 1993 as a whole-day event. Early information can be obtained from Davide Santachiara on +39 522 70409 or Eros Forenzi on +39 342 492323. The Italian Meeting is a regular for several international QL dealers, and they are very much hoping that Miracle Systems will be there with the QXL

QXL in USA

Miracle Systems shipped their first release models of the QXL QL-in-a-PC card at the ILQR Show at Newport, Rhode Island in America's New England on June 5th.

The full SMS/Q operating system, developed by Tony Tebby, was not fully complete at the time of shipping, but all the basic functions are in place and Miracle's Stuart Honeyball is expecting the remainder of the software to be despatched to users at the end of June.

At present the QXL floppy disk handling is up and running, so that the card is fully internally functional. Work is in progress on the print functions, the hard disk and parallel port functions, and the interpreter, which is being written by Laurence Reeves of Minerva fame.

Already several UK users have been able to load software up on the QXL during demonstrations and see it running. Interest in the QXL was "pretty good" in New England. The ILQR Show was attended by about 60 QL enthusiasts from all over the USA and neighbouring Canada (new England is "not far" - a mere 200 miles - from the Canadian border).

Miracle's Stuart Honeyball, himself no mean traveller, was struck by the distances covered by US visitors. He met one user from Albuquerque, and another who had driven the best part of 1,000 miles to Newport. One *QL World* reader came from Canada after seeing the announcement in the May issue.

The UK contingent were Stuart, Tony Firshman (TF Services) and Bill Richardson (EEC). At opposite ends of the high-tech scale, Stuart spoke highly of Virgin Atlantic's Super Economy service - "Most enjoyable flight, passable food and plenty of space" - and spent the Thursday of their arrival cycling round Rhode Island with Tony Firshman.

Miracle are themselves sponsoring a QL meeting - in Belgium, on 17 July 1993. The location will be the Eurovolley Hall, Beneluxlaan, Vilvoorde, Brussels. For more information, see next month's *QL Scene*, or contact **Miracle Systems on 0904 423986**.

Small Laser Printers

Rank Xerox introduced two new 'entry level' (small and low-cost) laser printers to its range in January. 'Small and low-cost' for a laser printer is in a different ballpark from printers of other types, but nonetheless laser printers are now getting the attention of serious users who need high-quality print output.

The smaller model, the Xerox 4010 II, includes HP Laserjet II (PCL4) emulation and costs £869 with one-year on-site warranty. The Xerox 4010 III has HP Laserjet III (PCL5) emulation, a 3-year on-site warranty and costs £1,095.

Both models operate at 4 pages per minute, 300

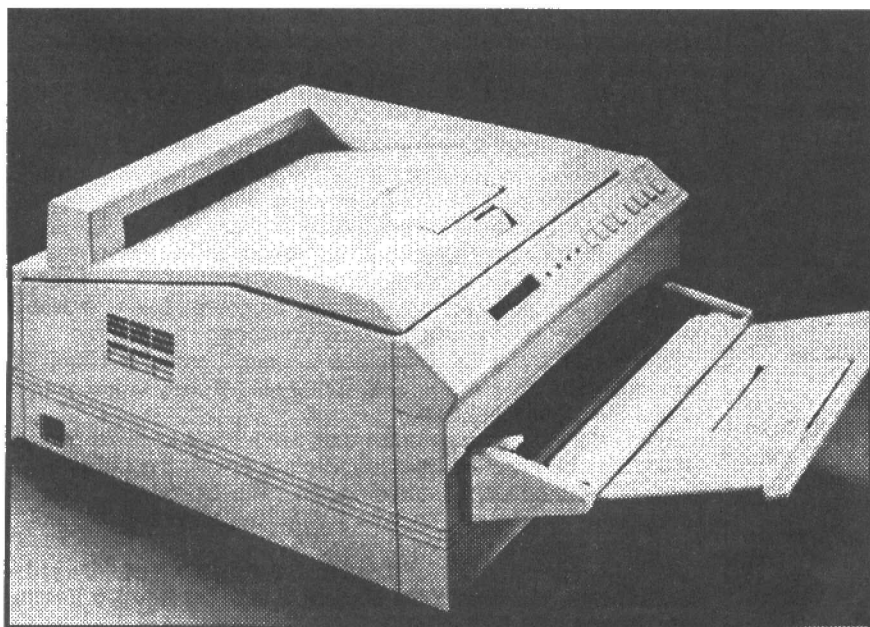
dots per inch and are designed to handle up to 5000 pages per month. The 4010 II has standard 512K ram. The 4010 III comes with 2.5MB of ram, and both can be upgraded to 4.5MB. There are also upgrade kits for other industry standard emulations. The machines will handle A4, US letter or US legal paper sizes between 60 and 120 gsm weight, as well as envelopes, transparencies and other non-standard stock.

Ink-jet printers have offered a lower-cost alternative to laser printers for good-quality output, although they do not match the versatility of a laser printer, and some users have found problems with

ink-handling.

The Hewlett Packard Deskjet is probably the leading inkjet range, used by many QL users with excellent results.

Information from dealers or:
Rank Xerox UK Ltd., Bridge House, Oxford Rd., Uxbridge, Middx UB8 1HS. Tel. 0895 251133.



QLScene

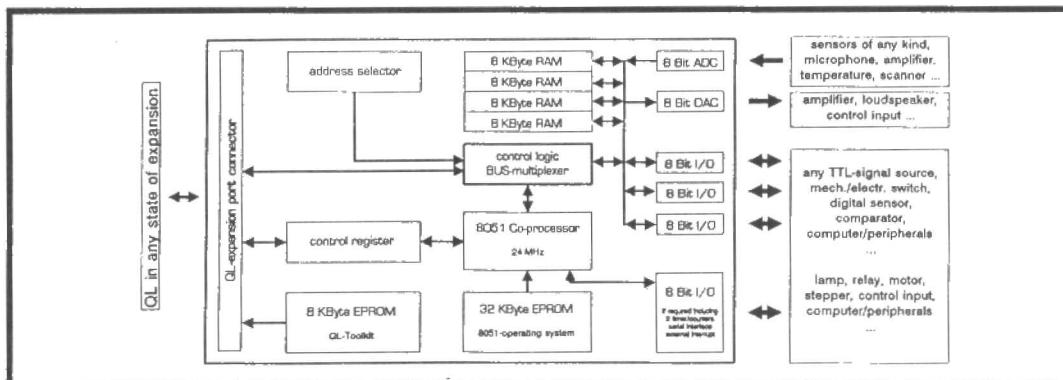
QLScene

New I/O from Falkenburg

W N Richardson (EEC) have their hands on a new QL multiprocessor input-output interfacing system from German developer Jurgen Falkenburg. The IIO-Card has an on-card integrated 80C51 24-MHz co-processor, a user-friendly Basic Toolkit, and its own dual-ported 32K static ram, which can be used for programming in addition to the QL's own commands.

Falkenburg describes the Card as being switchable to be compatible with any QL system, including the Gold Card. He suggests scanner, sound-sampler, oscilloscope, logic analyser, model controller and frequency generator as just some of the types of function which the IIO-Card can be applied to.

A preliminary product leaflet with further information can be obtained from **W H Richardson & Co., 18-21 Milsbourne House, Chiltern Hill, Chalfont St Peter SL9 9UE. Tel. 0753 888866.**



Extended Spectra

Simon Goodwin

The full *Speculator* release is at last complete, and available from Qubbesoft, SJPD and the Quanta library. This freely-distributable 48K Spectrum emulator has been extended since the review in April's *QL World*.

Fast machine-code routines to convert programs and screens into QL format replace the original C and SuperBasic code, and dozens of Toolkit commands have been added to make it easier to manipulate ZX programs and data files from Qdos.

Compatibility is improved now that 104 un-documented Z80 opcodes have been implemented by emulator author William James. Dave Barker of Charvelsoft has extended QSPEC to load cassette files on any expanded QL, automatically adjusting its timing routines to match the system's ram speed.

You can now save as well as load tapes through the QL NET port, and even headerless files and hyperloads are supported. Hackers will be pleased to hear that Z80 register values are available to SuperBasic, and the configurable disassembler can generate labels automatically.

Dave Walker's *Sectorkit* allows QL programs to read and write data on Spectrum and One Per Desk microdrives, as well as obscure disk formats. A new conversion utility handles snapshot files from the fast PC emulator JPP, the Amiga Spectrum emulator, and the Spectrum's own Mirage Microdriver. All these are included on the 720K *Speculator* 93 disk, together with complete 68000 source code for QSPEC, the emulator and disassembler.

Meanwhile in the Netherlands Carlos Delhez has extended his shareware emulator *Spectator* to impersonate the 128K Spectrum, although this has not yet reached QL World. Ergon Development of Italy has demonstrated another emulator for 128K Spectrum programs, based on ZM/2 but extended to support the extra roms and 128K ram paging. I have tested the prototype, and find that it works, although the paging makes it slower than other Ergon emulators. For further information contact **Ergon boss Davide Santachiara on 010 39 342 492323.**

New developments from Ergon

Ergon Development has introduced several new programs into their roster of QL utilities and 68000 Spectrum emulators. Following ZM/2, ZM/3 and ZM/hT (see Simon Goodwin's article in the March 1993 *QL World*) there is now ZM/128, a Spectrum 48K/128K emulator which also implements Interface-1 emulation. The Supervisor is now similar to ZM/hT, with joystick emulation. ZM/128 supports the new Z80 version 2 snapshot format, including utilities to convert 128K Disciple snapshot.

Ergon's benchmarks show that 128K emulation is a little slower than 48K emulation, but gives good speeds with the Gold Card. ZM/128 can be bought as part of the other emulator packages for an additional 20,000 Italian Lire (about £13). "ZM/x" (ZM/2 and ZM/3 together) now costs 60,000 ITL, and ZM/hT now costs 90,000 ITL (plus post and packing, and the extra 20,000 ITL if ZM/128 is required).

Ergon's *Floppy Disk Utilities* is a stand-alone program comprising a disk editor (search, collect, recover, etc.), disk copier/verifier (single and dual drives, with many features), and handles DD, HD and ED disks, as well as "alien" disks (PC, Spectrum, etc.). FDU costs 35,000 ITL plus post and packing.

The *DEA* Intelligent Disassembler is now into version 5.10 Plus 2, with Qdos/SMS/Wman/Pointer Environment keys automatic remarking in the output code. System and Basic Variables keys are recognised. The price remains at 55,000 ITL plus P&P.

Joining in the move towards cheap demo disks, Ergon issue a disk with over 1.5MB of compressed data, information, and public domain version demos of many of their programs. This is available on 3.5in disk for six International Reply Coupons.

Ergon's post and packing charges for normal orders is 12,000 ITL (about £5) per order (not per item). For very large/bulky orders, please contact them for information.

Davide Santachiara also notes that direct bank transfers can be made to Banca Popolare Dell'Emilia Romagna Italy, Swift BPOIT22, (Telex 510031 emipop), Sede Reggio Emilia CC 6533/73 Davide Santachiara. This requires an extra 10,000 ITL added to the order, as this is what the bank charges Davide.

All correspondence to **Ergon Development, Davide Santachiara, Via Emilio De Marchi 2, 42100 Reggio Emilia, Italy. Phone +34 522 70409.**

Very Basic SuperBasic

Dilwyn Jones loops the LOOP in SuperBasic.

In this third part of SuperBasic for new programmers, I'll be looking at looping structures. That's rather a big term to use at the beginning, so let's simplify it a little.

A loop is a part of a program which performs a process over and over again - hence the name. Or, as we say, can be executed over and over again. This saves writing identical bits of a program several times to do the same job.

The simplest type of loop is one which just goes blindly round and round, never stopping. This can be written using a REPEAT loop. REPEAT is a command which tells the computer to perform the following line or lines of Basic until told to stop. This is the simplest example:

```
100 INPUT "Enter your
name: "; n$
110 REPEAT printing
120 PRINT n$;
130 END REPEAT printing
```

All that this does is ask you to enter a name, and then keeps printing it on the screen until you press BREAK (Ctrl and Space held down together) to stop the program. The part between REPEAT and END REPEAT is carried out well - repeatedly. Note that each REPEAT loop has a name (in this case "printing"). This name can be used as an ordinary variable and you can give it a value if you wish, though most people just use it to identify

where the loop starts and ends. Names are very necessary, because you can have several loops one inside the other, a process called Nesting.

If you think it is rather pointless not being able to stop the program in any other way, you'd be right! There is a keyword, EXIT, which can stop executing what's in the loop and jump out of it to carry on after the END REPEAT statement. EXIT is not much use by itself, but once we've discussed the IF...THEN clause, you'll see how useful it can be. REPEAT loops are generally used where a terminating condition is not known at the time of entry to the loop.

FOR/END FOR

The second type of loop is a FOR/END FOR loop. This allows us to make a program loop which is controlled by a variable which takes a preset value or range of values. The value of the variable is automatically altered each time the program goes round the loop.

The start and end values are set at the start point of the loop.

```
100 FOR a = 1 TO 3
110 PRINT "*"
120 END FOR a
```

This means that "a" is to have the values 1, 2 and 3. So the part between FOR and END FOR is executed three times, printing three asterisks. The program encounters the

FOR a statement and realises that it must perform the part in the loop several times, starting off where a has the value of 1 and keeping going until a has been 3. The first time, a is 1. An asterisk is printed. The program then runs into the END FOR statement, so it jumps back to the FOR a statement and works out the next value of a, which is 2. It checks that the loop has not reached the last value and prints another asterisk. Again, it jumps back from the END FOR to the FOR a statement and updates the value of a to 3, prints an asterisk and does the same thing again. But this time, the loop counter (as "a" is called) becomes 4, which is past the range specified, so the program now knows it has finished that loop and jumps to just after the END FOR statement. Since there is nothing there in this example, it realises it has finished the entire program and stops.

In the example given, the variable "a" changes its value in steps of 1, which is what always happens if you don't specify a value-change increment.

The keyword STEP allows you to change this value. It can be a whole number, or a decimal fraction, positive or negative, so it is very flexible.

```
100 FOR a = 1 TO 6 STEP 2
110 PRINT "**";
120 END FOR a
```

Here the value of "a" starts at 1, the next time round it becomes 3, and finally becomes 5. Since the next value would be 7, and beyond

the range specified, the loop would end when the value reached 7 (so only three asterisks would be printed in this case).

If you wanted to make the loop count backwards for some reason, you can do so by arranging the values in suitable order and using a step value which is negative. This example prints an asterisk which appears to move from right to left across the screen by printing the asterisk in different positions, using a slight pause (PAUSE is a command which tells a program to wait for a given time, which is expressed in units of 1/50th second or 1/60th second, depending on which model of QL designed for which country you have) to allow the symbol to be displayed for long enough to see before being wiped again by printing a space over it and printing it again in a different position to give the appearance of movement (albeit rather jerky movement).

```
100 CLS
110 FOR a = -20 TO 1 STEP -1
120 AT 5,a : PRINT "*"
130 PAUSE 5
140 AT 5,a : PRINT " "
150 END FOR a
```

There is still more we can do with FOR. Instead of just having one range of values, we can have a list of individual values and ranges, like this:

```
100 FOR a = 1,2,4 TO 8 STEP
2,15 TO 10 STEP -1,20
110 AT 10,a : PRINT "*"
120 END FOR a
```


In this case it executes the range of values between the commas one at a time, finishing one set before moving on to the next. Each range can have a STEP value if you want one. This is useful to place items in a list of positions. Normally, you'd find that most FOR loops are tidy and use just one range of values for most purposes.

Nested loops (loops one inside another) are useful for creating lists or for printing in different directions, for instance, this little program prints a box of asterisks:

```
100 FOR y = 1 TO 10
110 FOR x = 1 TO 10
120 ATyx
130 PRINT "*"
140 END FOR x
150 END FOR y
```

In that example, the "y" loop steps down the screen from one line down to 10 lines down. For each line, the "x" loop prints a row of 10 asterisks across.

There is another keyword called EXIT (which allows us to make a program jump out of a loop) which we should discuss here, but since we need to know about IF...THEN structures to make decisions so as to know when to leave a loop, we'll look at those first.

Computers are good at making strict decisions in simple cases, no beating about the bush like humans! We might take a long time to make up our minds but computers can, if the information is available, make a quick and simple decision. An example of a human decision is to go out if it's nice weather, or decide to stay in when there is bad weather (ignoring silly people who seem to insist on climbing tall mountains in all weatherst).

We might write this in Basic as something like:

```
IF weather = fine THEN
GO_OUT
ELSE
STAY_IN
END IF
```

In other words, the computer tests the condition after IF and if it works out to be true or valid, it executes the part of the program between IF and ELSE. If the weather was bad, it would be a false statement (a valid condition is normally referred to as TRUE and an invalid condition is usually referred to

as FALSE) and would therefore execute the part between ELSE and END IF.

So we can see that an IF clause can provide two separate courses of action as a result of making a decision. In fact, it can be made to provide several courses of action by having more than one IF clause.

```
100 x = RND(1 TO 2)
110 IF x = 1 THEN
120 PRINT"One"
130 ELSE
140 PRINT"Two"
150 END IF
```

That example showed a simple two case IF clause. Quite simple, because x is given a value of either 1 or 2 by the RND statement. It is possible to write more complex versions for more than one value:

```
100 x = RND(1 TO 3)
110 IF x = 1 THEN
120 PRINT"One"
130 ELSE
140 IF x = 2 THEN
150 PRINT"Two"
160 ELSE
170 PRINT"Three"
180 END IF
190 END IF
```

I hope that last example is clear to you - basically it tests for two of the three possible values and if it is neither of these it must be the value 3. That example could be written in another form, which is shorter but which would not be good practice in large programs:

```
100 x = RND(1 TO 3)
110 IF x = 1 THEN
PRINT"One"
120 IF x = 2 THEN
PRINT"Two"
130 IF x = 3 THEN
PRINT"Three"
```

You will notice that there is no ELSE and no END IF in these clauses. This is where things get a little more complex! There are short form versions of IF, FOR and REPEAT loops just for when you want to do something in a single line.

In this case, the action to be executed is placed immediately after the THEN keyword. If you want an ELSE statement as well, place a colon before it, like this:

```
100 x = RND(1 TO 2)
```

```
110 IF x = 1 THEN
PRINT"One" : ELSE
PRINT"Two"
```

The keyword THEN can usually be left out of an IF statement. This causes no problems in the long version, but in a short version if you try to enter something like IF x=2 PRINT"Two" it gets refused as you try to type it in, the computer gives a "Bad Line" report. You can get around this by putting a colon where the THEN would have been, so the statement would become IF x=2 : PRINT"Two"

The colon is used quite commonly to separate two or

with multiple statements, it becomes really unclear and when you go back to look at it some time after you've written it, you may not even understand your own work if it's an untidy, badly laid out mess, so think tidy!

In the short forms of FOR and REPEAT, the colon is essential. Here's an example using FOR.

```
100 FOR x = 1 TO 40 : PRINT
***;
```

Note how much shorter this is than when written as one line containing a FOR statement, another containing a

```
100 REMark Guess My Number
110 CLS
120 PRINT'Computer is thinking of a number,'
130 PRINT'which is from 1 to 100 (inclusive). '
140 PRINT'Please try to guess it by entering'
150 PRINT'a number, you will be given clues'
160 PRINT'as to your accuracy.'
170 LET tries = 0
180 LET number = RND(1 TO 100)
190 REPEAT program
200 INPUT'Your guess ? ' ; guess
210 LET tries = tries + 1
220 IF guess = number THEN
230 PRINT'Congratulations! It took ' ;
240 PRINT tries ; ' tries.'
250 EXIT program
260 ELSE
270 REMark it was not correct, so give clues
280 IF guess < number THEN
290 PRINT'Too low.'
300 ELSE
310 PRINT'Too high.'
320 END IF
330 END IF
340 END REPEAT program
350 PRINT'Program Finished.'
```

more commands on the same line. These are called, not surprisingly, "multiple statements". Too much use of these can make a program difficult to follow when someone else tries to understand your program, but for short lines it is quite useful, especially where related commands such as AT and PRINT are used together:

```
100 AT 5,5 : PRINT "Hello"
```

In these cases, it actually simplifies a program, but you must use it with care. When you start to write more complex programs, if you fill a program

PRINT statement and another with the END FOR statement. Not only do we save having to write three lines of Basic, we can also leave out an END FOR statement, because once the computer reaches the end of the line, it automatically assumes that there is an END FOR statement there (you can include one if you wish, but as long as there is only one FOR statement on the line, it is not essential).

We can also do the same with a REPEAT loop:

```
100 REPEAT printing :
PRINT"Hello ";
```

Like a FOR loop, it does not require an END REPEAT statement at the end of a line in this case.

I mentioned the EXIT statement earlier. When used in

```
170 END REPEAT
enter_numbers
```

It is not quite so easy to explain the useage in a FOR loop, but I will try. It can be

```
100 REMark Executive Decision Maker
110 CLS : PRINT'DECISION MAKER'
120 PRINT'Think of a question and press ENTER.'
130 INPUT a$: REMark a followed by dollar symbol
140 PRINT'I have made the decision for you.'
150 PRINT'My suggestion is: ';
160 LET random_number = RND(1 TO 3)
170 IF random_number = 1 : PRINT 'YES'
180 IF random_number = 2 : PRINT 'NO'
190 IF random_number = 3 : PRINT 'MAYBE!'
```

conjunction with an IF statement it gives you the possibility of jumping out of a loop when a certain condition has been met, such as this little program which jumps out of a loop when the random number generated is a 6

```
100 REPEAT loop
110 number = RND(1 TO 10)
120 IF number = 6 THEN
EXIT loop
130 PRINT number
140 END REPEAT loop
```

The program keeps going, printing the random numbers generated until it has thrown up a number 6, when it jumps out of the loop (jumps past the END REPEAT statement).

Using EXIT to jump out of a FOR loop is also allowed, though it is not used as often as it is in a REPEAT loop. An EXIT statement in a FOR loop will cause the program to jump to just beyond the END FOR statement.

Finally, there is one more keyword associated with these loops, this is NEXT. This simply causes the program to go back to the statement which started the loop, allowing it to loop around from a different point if special circumstances dictate in your program. Its use is obvious in a REPEAT loop, where it could be used to get around entering problem values for example:

```
100 REPEAT enter_numbers
110 INPUT'Enter a positive number: ';number
120 IF number < 0 THEN
130 PRINT'Negative numbers not allowed.'
140 NEXT enter_numbers
150 END IF
160 PRINT'OK': EXIT
enter_numbers
```

Here, a loop is set up to enter six numbers. If you wish to stop the program after entering less than six numbers (for instance, if you realised you'd made a mistake and wanted to start again) simply enter a value of 0, which is a signal to the program to quit. If all goes well and you enter all six numbers, the program prints the message in line 160, but if you enter 0, the program jumps out of the loop called "a" and ignores the bit between NEXT and END FOR. You will be pleased to know that "loop epilogues" are not used that often, though it is useful to know of the technique in case you find it used in someone else's program!

Examples

After all that theory, let's type in a couple of short listings to put the theory into practice. First of all, a short game based

on decisions. This is Guess My Number, one of the oldest games on a computer, but short and easy to understand.

The computer generates a number at random from 1 to 100 and invites you to guess what it was. You should type in a number from 1 to 100 when asked (remember to press Enter after the number). The computer will congratulate you if you get it right, or give you clues until you get it right.

Figure two is another old favourite, the Executive Decision Maker. If you are feeling tired and stressed and can't make your mind up, let this program do it for you! It gives a 'yes' or a 'no' or even an occasional 'maybe' for the really indecisive!

Decisions!

Why not modify this program by adding a FOR loop around it to allow it to make more than one decision for you? Just don't blame me if you really do make decisions based on the dreaded QL random number generator!

Figure three allows you to

test your luck by playing Russian Roulette! If you have luck like me, you will consistently find yourself dead. It shows how to use multiple IF statements to make a decision which can have more than one outcome. In this game the gun has six chambers. You fire it at yourself a given number of times depending on how lucky you feel! If the chamber is empty, you survive that shot and the gun just clicks, but if it bangs it contains a bullet and you are dead! This routine has IF statements all over the place and a FOR/NEXT/END FOR loop (a FOR loop with an epilogue as described above). It also adds a RANDOMISE statement which means that a new 'base' number for the random number generator is set, hopefully ensuring that they will be different each time! Note the use of the extra random number once the program has 'found' the chamber with the bullet to give you a one in ten chance of surviving with a freak occurrence of a jammed gun (line 310). Note how the FOR loop counter value (the value of the variable "x") is used to choose a comment for the shot number.

```
100 REMark Russian Roulette
110 RANDOMISE
120 CLS : PRINT'The gun has 6 chambers.'
130 PRINT'One contains a bullet.'
140 PRINT'How many times would you like to'
150 INPUT'risk your life ? ';shots
160 CLS
170 LET bullet = RND(1 TO 6)
180 LET random_number = RND(1 TO 6)
190 FOR x = 1 TO shots
200 PRINT'Shot ';x;'. Press ENTER.'
210 INPUT a$
220 IF x = 1 : PRINT'The tension is high...'
230 IF x = 2 : PRINT'Getting risky...'
240 IF x = 3 : PRINT'Surely this time...'
250 IF x = 4 : PRINT'No chance...'
260 IF x = 5 : PRINT'You must be plain lucky...'
270 IF x = 6 : PRINT'You can't survive the sixth shot...'
280 LET random_number = random_number + 1
290 IF random_number = 7 THEN LET random_number = 1
300 IF random_number = bullet THEN
310 IF RND(1 TO 10) = 1 THEN
320 PRINT'How lucky can you get!'
330 PRINT'That chamber held the bullet,'
340 PRINT'but the gun jammed.'
350 ELSE
360 PRINT'BANG! You're dead! R.I.P.'
370 EXIT x
380 END IF
390 ELSE
400 PRINT'CLICK. You survived that one!'
410 END IF
420 PRINT
430 NEXT x
440 PRINT 'Good grief, you survived all ';shots; ' shots!'
450 END FOR x
```

QL SERMouse

Bryan Davies scuttles through three versions of an adaptable QL mouse package.

Programs written with a mouse in mind can be awkward to use without one. This is very evident with those programs which use the QJump Pointer Environment. While it is possible to access the normal commands through the keyboard, it can be rather long-winded to do so. It is both regrettable and surprising that the QL has never benefited from a "standard mouse". There has been the Ice mouse, the ABC one, the Smiling Mouse, the Sandy mouse, the Qimi mouse, and various others. The Ice one worked well, but was limited to Eidersoft's programs, the ABC had a variety of mechanical and software problems, the Smiling Mouse showed promise which was ruined by software that was never finished and marketing that left various people disappointed. The Qimi mouse (and interface) should, perhaps, have been "the one", coming, as it originally did, from the producers of QRam and QPac; the Qimi interface is now available again from the Quanta users' group, some problems having apparently been sorted out.

SERMouse is a software mouse driver from Albin Hessler Software (in Germany) that is presumably intended to fill the gap the Qimi interface left when it originally went off the market. It is clearly designed to work with the Pointer Environment. One big merit is that standard mice from other computers can be used with it. The instructions do not specify particular brands and models of mouse that are safe to attach to the QL, but both the UK suppliers (W N Richardson and Software 87) are shipping a mouse and connecting cable with the

software, relieving the buyer of a few problems.

Variables

To avoid confusion, it should be explained before going any further that the items received for review came from different sources, and the review was done in three stages:

1) Connecting a standard PC-type mouse to the QL. This involved reading the SERMouse instructions and soldering two connectors to a cable, to make an adapter cable to go between mouse connector and Serial port connector.

2) Using the mouse software supplied by Software87 (with update from Albin Hessler), and the mouse and adapter cable as mentioned in 1). Note that Software87 normally supply a complete kit - mouse, adapter cable and software - and the information given below on making adapters for a mouse is for the benefit of anyone buying the software on its own (eg direct from Albin Hessler Software). The mouse referred to here is a standard PC type, with three buttons, connected to the serial port by cable.

3) Using a kit comprising mouse, adapter cable, mouse software, and drawing software, supplied by W N Richardson. The mouse referred to here is an infra-red type, which has no cable itself but transmits to a receiver unit connected to the QL through its cable and an adapter. The same supplier offers a kit with standard, cabled mouse for £10 less than the infra-red version; it is assumed the standard kit should be

essentially the same as that offered by Software87.

Connecting a standard mouse

Many users would not feel like chancing their money and QL by plugging in any old mouse. The decision on what mouse to use was made easier by the information from Software87 (and in the SERMouse instructions) that a standard 3-button PC mouse - such as the Genius GM-6 which I happened to have spare - would work. 2-button PC mice also should work, although mice that perform auto-sensing to switch between Microsoft (2-button) and Mouse Systems (3-button) modes may not work. Potential buyers of the complete package need not concern themselves with some of the following comments, as the latter are made for those who intend to find their own mouse and cable.

A note of caution in the SERMouse instructions was a bit worrying - mice require 5 volts DC, but the QL serial ports only supply at 12 volts. It is stated that the load of a mouse will drop the 12V down to 5V, making the supply safe for use with mice. Certainly, the voltage from the serial port checked dropped to about 5.5 when only a mouse was connected, and the mouse behaved without tantrums. When a printer is connected to the other port by a direct serial connection, there is no additional load on the supply and there should be no trouble with mouse function but, when a serial-parallel converter is used - as one commonly is on QL systems, because printers normally have parallel interfaces - the combined load on the 12V supply is likely to be too great for correct operation (of either device).

Loading fix

The suggested course of action to get around this difficulty is to fit an individual 680 ohm resistor to pin 6 on each serial port, from the +12V line on the motherboard. One such resistor is fitted as standard, but to both ports together. The instructions on this point were not clear enough to make one happy, and average users would not want to tamper with the QL internally. Effectively, this reservation appears to rule out this form of mouse interface (with either standard or infra-red mouse, presumably) for many users, which is a pity. However, it is stated that the mouse or a printer serial-parallel link can be used separately without problem, so the user has the (inconvenient) option of unplugging one or the other. It is made clear that 12V should not be applied directly to the mouse, presumably meaning that a stronger 12V supply than that in the QL could feed the mouse with well over the specified 5V and, thereby, make it roll over. The last section in the instructions, headed "Guarantee", basically says "be it upon your own head", and one cannot blame the program writer for that. It is not his fault that interface wiring is a bit of a minefield. The user with reasonable DIY capability need not be put off by this; a standard QL and PC mouse should work together, using a cable wired to one of the four given configurations (my own set-up uses the one for the 6-connection 25-pin D mouse and 9-pin D SER2 port).

Another big plus-point for SERMouse is that no form of interface board has to be attached to, or inserted into, the QL. When the full kit is bought, all the user has to do with the hardware is plug it in, but it is important to advise your supplier

of the type of SER connector on the QL concerned. For the DIY purchaser (of the software alone), you first have to find a way of linking your mouse to your QL, and that is not as straightforward as one would like it to be. This is not the program-writer's fault, though. The instructions provide four tables, covering eight possible cable configurations. This is not as bad as it may sound, since the configurations split into groups which can all be identified easily - except one. It is obvious whether your QL has 9-pin D-shaped connectors or 6-pin PCC ("telephone") sockets for SER1 and SER2. Equally, the number of pins on the D mouse connector - 9 or 25 - will be clear. You are free to use SER1 or SER2 (remember that they are wired differently), and the

wiring groups listed applies to your mouse; the tables (see box) show 9- and 25-pin mouse connectors using only 4 of those pins each, but they also show a variant of 9-pin connector with 5 pins used and a 25-pin variant with 6 pins used. Fortunately for me, I had long ago checked the serial port wiring on both my UK and US QLs (PCC and D connectors, respectively), making it relatively easy to make-up a suitable cable for the US QL normally used for my reviews. You may be able to get a ready-made cable (try TF Services), but you will still have to specify the wiring configuration required.

There is always a question mark when one has been wielding the soldering iron, but things went fine with this job, and the mouse worked properly

With serial ports

The pin connections given here are for the connectors at the ends of the linking cable. "D9" indicates a 9-pin D-shaped connector, as used on German- and US-market QLs and on current PC ("PS/2") mice. "PCC6" indicates the 6-pin telephone-style plugs and sockets used on UK-market QLs. "D25" indicates the standard RS232 connector used on PCs and mice until recently. "M" and "F" indicate Male and Female, respectively. See table one.

Using the mouse

Mouse-pointer movement across the screen is smooth, and response to the buttons is clean. Movement from side to side of the screen required less than half of the width of a

without the PE itself being loaded, but the behaviour is rather uncertain and is more a curiosity than a useful feature. The fact that these programs - for example, Files 2 (from TaskMaster) and Quill - respond at all to the mouse suggests that it would be possible to make them properly "mouse aware". This raises a prospect that is definitely interesting, as there must be many QL users who would like to have just one mouse, that could be used to control most, if not all, of their favourite programs.

Whatever the current level of attention to the Pointer Environment, it is not at all likely that it will become universally-used within the QL community, and it is desirable that SERMouse be made to work with existing non-PE programs.

Clearly, Psion are not likely to develop their Quartet to use the mouse, but it is not beyond the bounds of possibility that someone else might introduce the necessary links into these programs, rather in the manner that Turboquill made the latter program so much more usable some years back.

Instructions

The SERMouse instructions run to 13 A5 sides, with a further four sides on SuperBasic extensions on the supplied disk (which do not relate directly to the mouse). The supplied 3.5-inch DD disk contained 22 files (half of them duplicates), which were in the bi-lingual format that seems normal with German PE-based programs. An example Boot routine is supplied, and this makes it fairly obvious what order to put the various vital ingredients of your own system Boot routine in. The only file which is actually essential is SERMouse itself, and that has to be loaded with the LRESPR or RESPR command, after PTR_GEN, WMAN and HOT_REXT, if they are in your Boot (they are not supplied on the disk). You are advised on how to create a usable version of SERMouse, but the supplied one worked without any need for

decision on which one to use will probably be made for you by the fact that a printer is connected to SER1. SERMouse comes configured for SER2, which makes good sense.

Wiring groups

What may not be so easy is determining which of the two

with the first Pointer program tested (QSpread). When one has used Abacus, a program like QSpread feels cumbersome if only the keyboard is available; the presence of the mouse made it much nicer to user, a two-handed technique being desirable to do the necessary PE manipulation and the data entry.

normal mouse mat, which is quite reasonable. The resolution can be decreased, thereby decreasing the movement required, with the supplied configuration utilities.

The mouse functions with non-PE programs, with or

MOUSE D25 M		QL D9 M		QL PCC6 M	
CONNECTOR PIN:	QL SIGNAL:	SER1:	SER2:	SER1:	SER2:
2	TxD	3	2	3	2
3	RxD	2	3	2	3
4	+12V	9	9	6	6
7	GROUND	7	7	1	1

OR:

1	GROUND	1	1	1	1
2	TxD	3	2	3	2
3	RxD	2	3	2	3
4	GROUND	1	1	1	1
7	GROUND	1	1	1	1
20	+12V	9	9	6	6

MOUSE D9 M		QL D9 M		QL PCC6 M	
CONNECTOR PIN:	QL SIGNAL:	SER1:	SER2:	SER1:	SER2:
2	TxD	2	3	2	3
3	RxD	3	2	3	2
5	GROUND	7	7	1	1
7	+12V	9	9	6	6

OR:

2	TxD	2	3	2	3
3	RxD	3	2	3	2
4	+12V	9	9	6	6
5	GROUND	1	1	1	1
7	+12V	9	9	6	6

configuration, in the review system. When the PE files are not present, SERMouse works in "cursor mode", mouse movements being translated into cursor-key presses.

All this might seem unnecessary, when all you have to do is copy one file onto your Boot disk and put a load command for it into your Boot file. The rest deals with the various configuration parameters/commands. One thing about the QL that seems to bug programmers is the inability to set different Baud values (data-transfer rates) for the two serial ports. As SERMouse works at 1200 Baud, and most people will be using their printers at 9600 Baud, there is potential for conflict, but the mouse driver avoids this by setting the rate for both ports to 1200 initially, then suspending the mouse input and resetting the rate for the other port to its normal value, if a program requests a channel for output to a printer. The mouse input is then automatically re-connected when the print has finished. You can have printer and mouse active together, but at the expense of turning the printer data rate down to 1200 Baud. So, you do not do much mousing on a day when there is a heavy printing load. The Hermes replacement for the 8049 chip is stated to be compatible with SERmouse, allowing the two ports to run at different rates with no need for "operator intervention".

Psion problem

There is a rather more annoying bodge required if you use Psion programs, as they apparently do not automatically close channels opened for printing. You have to "manually" - that is, through SuperBasic - set the Baud value to 1200 to get the mouse going, or 9600 for print from the programs.

The remainder of the instruction booklet deals with configuration of SERMouse, using the standard PE Config routine, and explanations of the SuperBasic commands which can be used separately to alter the parameters. You can change just about anything - Baud, port, pointer and cursor speeds, acceleration factor, double-click delay, etc. A screen-blanking function is incorporated, with a

range of 0-20 minutes, with a 5-minute default. There is a detailed list of button-press combinations and their effects.

Infra-red mouse

The kit comes in a smart box, nicely presented but rather daunting. It appears to be designed primarily for PC users, with items needed by QL users added on. The mouse had the usual three buttons on top, plus an additional one on the left side; at the front was the "eye". Four disks were for use with PCs - no use for the free game, for which there was a substantial instruction book! The remaining 3.5-inch disk was the SERMouse software. The mouse-driver file was version 1.04, whereas 2.07 was supplied for testing the standard mouse; both versions were used when testing the infra-red mouse, and there was no apparent difference in behaviour, but it is to be hoped the later version and its (better) instructions will now be supplied with the kit mentioned here (which has been in circulation for some time). The 25-to-9 pin adapter that usually comes with PC mice was replaced by a 25-6 pin (PCC) adapter for the QL. The SERMouse instructions implied that SER2 was the expected connection, so that socket was used.

The instructions were similar to those provided by Software87, but less detailed. It was not clear what had to be done to get the software installed, although all that was required was to copy one file onto my system boot disk and add one line to the boot. It would not be obvious to some purchasers which file to copy and how to add the boot file line, and there were two driver files to choose from, neither of them mentioned in the instructions. The first of the files was tried and it worked; the other one was not tried.

Fitting

The mouse had its own instruction book, which was quite sufficient for the little work that was required. After the adapter had been fitted to SER2 and the infra-red receiver connector plugged into it, two AAA batteries (supplied for the review) had to be fitted into the mouse, a button on the side

then needed pressing, and the mouse was up and running. Leastways, it did what was expected of it when used with Jochen Merz's HyperHELP program.

From there on, behaviour was the same as with a standard mouse. To conserve the batteries, the infra-red mouse goes into "stand-by" mode after the mouse has not been moved for 5 seconds, and it switches off automatically after 20 minutes without use; the extra button (on the left side) is used to re-activate it. Presumably, then, you can leave the mouse for fairly long periods without the batteries going flat. The instructions state that the range of infra-red transmission is five feet, within an angle of 45 degrees either side of the direct line between mouse and receiver, and this claim appeared to be justified. You must not let anything get in between mouse eye and receiver, though - this device works at a frequency close to the visual range and pointer movement ceases once there is no line-of-sight connection. You can tell if the connection is good by watching the flicker of what looks like an LED in the window of the receiver; no flicker when the mouse is moved means the signal reaching the receiver is not large enough.

Same habits

Almost everything about using a standard mouse applies equally to the infra-red mouse. You would not notice the difference normally. From an operational point of view, the two mouse kits are essentially the same. The infra-red mouse will appeal to users who do not want a cable between the mouse and the QL. Bear in mind, however, that the receiver unit required by the infra-red mouse has a cable which is much the same length as that on a standard mouse, and the same type of adapter is required for the Serial port; that is, you have to find space for the same amount of connectors and cable, plus the receiver unit.

If you are in the habit of taking your tea on a tray, a few feet from your QL, you could still play

your favourite game, without simultaneously pouring coffee into the QL's delicate keyboard membrane. The maximum range would be about 8-9 feet (allowing for the length of cable attached to the infra-red receiver). Where space for the QL is limited, as it is on my workbench, the cable from a standard mouse can be a real nuisance. Two mice use the same mouse-pad on my bench, causing a considerable tangle of cable, and a certain amount of confusion for the other, occasional user! The infra-red mouse definitely has the edge over the standard one in such a situation.

The mechanics of the standard and infra-red mice are similar and you would need to clean the ball and rollers on both periodically. Neither one really requires a mouse pad to work on, but each might operate more consistently with one.

Conclusion

For using with Pointer Environment programs, SERMouse looks to be well worth having. Maybe some of the existing, popular programs will be adapted for use with this driver one day (but don't bank on it). The all-in prices quoted for the mouse/cable/software package are reasonable, bearing in mind what it would cost to have a cable specially made up. The normal shop prices for PC mice start at about £20 plus VAT, but you can get mice at half this price (or less). Above all, the package makes addition of a mouse to the QL a relatively painless experience, which is something that could not be said about several of the previous offerings. As the writer of the software commented, his approach may not be the best solution to the problem, but it is possibly the only one which provides something which is both cheap and easy to install. Experienced, regular users of application programs can usually get along faster with the keyboard, but less-experienced, occasional users would welcome the facility to click on menus containing meaningful words, rather than trying to remember the keypresses for commands.

INFORMATION

Program: SERMouse 2.07 (kit of software, mouse and adapter cable)

Price: £49 (standard mouse) £55 (infra-red mouse), £45 (standard mouse), plus £5 or £9 P&P.

Suppliers: W N Richardson & Co., 18-21 Misbourne House, Chiltern Hill, Chalfont St. Peter, Bucks SL9 9UE. Software87, 33 Savernake Road, London NW3 2JU.

Living with Hermes

Another view of the Hermes ILP, from Rich Mellor.

Hermes is a relatively new product for the QL which aims to cure various problems with the QL's 8049 intelligent peripheral controller chip. The IPC chip is situated just to the left of the two microdrives and controls keyboard input, sound, and serial input (although not serial output which is controlled separately). The 8049 contains some poorly-designed code, which causes problems in all these areas. The main problems are poor handshaking on serial input, leading to loss of some characters; keybounce; unreliable handling of serial input at baud rates greater than 1200 baud; and 'serial overrun'.

Serial overrun occurs when data is being read from the serial port; the 8049 seems to hold onto some characters and then release them all at once when the next character is read. This leads to extremely unreliable serial input, and the only cure would seem to be to turn the computer off and on, and start all over again (resetting the QL does not always clear the problem).

Predecessors

There have been other versions of the 8049: a later version from Sinclair Research and another independent release. Both of these only addressed the keybounce problems, and the latter actually introduces additional problems. Hermes sets out to fix all the old problems, and at the same time provide additional flexibility to the QL.

With the chip, you receive a seven-page manual and a disk containing some machine code extensions to access the additional utilities provided by Hermes. Oddly enough, the instructions for fitting the chip are at the back of the manual (although there is a contents page to direct you to the right place). If you follow the instructions, fitting the chip is relatively easy - you need to undo eight screws on the bottom of the QL and take it apart. Having located the 8049 chip, you then lever it out of its socket using a small flat-bladed screwdriver.

However, this is not easy because one end of the socket is very close to the edge of the QL case, so that the screwdriver can only be inserted at one end, making it all too easy to bend pins on the chip. Removal would be easier if a chip extraction tool was supplied, but with care and patience, the 8049 can be removed without breaking any pins. Hermes is then slotted into the socket, given a firm push home, and the QL put back together.

If you have anything else inserted into the 8049 socket, there is no need to worry as Hermes is exactly the same size as the original 8049 chip.

Key click

Once in place, the QL can then be switched on - if there are any problems, then you will not get the F1..F2 start-up screen. The QL should now operate as before, however, the first thing which you will notice is

the little click whenever a key is pressed. Reading through the manual, I see that this click can be turned off either by using the command `IPCEXT 6` (after loading the supplied machine code extensions) or by pressing the keys `Ctrl-Alt-Esc-4`. Unfortunately I cannot recommend this keystroke, as my full-sized keyboard makes it very difficult for the QL to recognise that another key is being pressed at the same time as the Esc key (or that two letter-keys are being depressed, for that matter). This is however a problem with the Jurgen Falkenberg keyboard interface (or possibly a PC keyboard) and therefore I cannot blame TF Services.

The keyclick is automatically enabled when the QL is switched on, but Minerva users will be pleased to note that if they turn it off, when Minerva is reset with a soft reset, then the keyclick remains disabled. Unfortunately, Hermes cannot produce a keyclick when a key is being auto-repeated (ie when it is held down to produce the same keystroke several times) as this is handled internally by the QL. Keyboard input is inevitably slightly slower with the keyclick enabled, but interestingly it also seems slightly slower where the keyclick has been turned off. This may of course be due to the extra code which is needed to check whether or not to produce the keyclick.

While on the subject of keyboards, I was pleased to see that Hermes improves key rollover, because since I installed my replacement keyboard, I have found it difficult to play games on the QL. This was because key rollover did not work very well at all (it was much better on the original QL keyboard), so that if I was pressing two keys at the same time (for instance, to move diagonally), if I released one key, the QL failed to recognise that the second key was still being depressed (forcing me to release both keys and then press the second key again). With Hermes, this problem is resolved, and I can happily play my favourite games once again (which by itself makes it worth buying Hermes!).

Baud rates

One of the most impressive enhancements is the ability to have different input baud rates on both `ser1` and `ser2`. The baud rates can also both be different from the output baud rate. The new function `RXBAUD%` is used both to set the baud rate for either `ser1` or `ser2` and read the current baud rate. Unfortunately, there is no way of reading the baud rate on one serial port without actually setting it, so you would need to use the following small function to do so:

```
100 DEFine FuNction  
TEST_BAUD(ser_no) 110
```



```
port=128*(ser_no-1)
120 rx%=RXBAUD%(port)
130
reset_baud=RXBAUD%(rx%&
&127+port)
140 RETURN rx%
150 END DEFINE
```

This should then be called by `x=TEST_BAUD(1)` for ser1 or `x=TEST_BAUD(2)` for ser2. The parameter returned by this function will then contain a value, the bits of which can be examined in order to determine if the given port is open or closed, whether it is affected by the BAUD command, and the current input baud rate used on that port. The manual explains the significance of each of the bits.

This extension to the serial driver allows you to read data from a modem attached to ser2 at 300 Baud, while still being able to access a printer at 9600 baud on ser1, without needing to alter the baud rate each time you swap between the two.

New Basic

Other new Basic commands allow you to read and set extra input and output lines which will enable you to connect various items to Hermes. At the moment, the only utility I am aware of which uses one of these output lines is the Qview CAPSLED kit, which has to be attached to one of the pins on either the 8049 or Hermes, and then lights up when Capslock is enabled (or Ctrl-F5 pressed). A small diagram showing the location of these pins on the chip might help, but this is really for hardware developers. There is also a function which can be used to test if Hermes is present (as well as returning the version number of Hermes) and one which allows you to read all 128 bytes of memory contained in Hermes (only six addresses are currently of interest to anyone, and these can mostly be accessed by the other commands).

Machine code programmers who wish to use Hermes facilities can do so through the MT.IPCOM trap, which allows you to access either the 8049 or Hermes. The MT.IPCOM

command only supports a maximum of fifteen commands, and on a standard QL, all of these commands are actually used by the 8049, although many books on the QL operating system do not list all of these. The manual explains that Hermes has to be accessed by first using the MT.IPCOM command \$9 (Microdrive Reduced Sensitivity - a command which actually did not work on an 8049) with a parameter of \$9 to switch on the Hermes extra command set.

Having done this, you can then check if Hermes is present by using the MT.IPCOM command \$F (Test), which returns the complement of the byte sent if Hermes is present (on a standard 8049, I believe it returns the same byte that was sent). Provided that Hermes is present, you can then access its utilities by using the Microdrive Reduced Sensitivity command with various parameters (all of the other commands remain active, with their original meaning) - the Hermes command set is only switched off once you have accessed one of its utilities using the Microdrive Reduced Sensitivity command.

Programming

All of this would make sense to a machine code programmer (honestly!) if only the QL books contained more information on how to use the MT.IPCOM command, as the actual syntax is extremely complex. The only books I have come across which contain enough information on the syntax are Andrew Pennell's QDOS Companion and the QL Technical Manual, although even these do not explain how you are supposed to use the Microdrive Reduced Sensitivity command (presumably as it never worked!). Luckily, you can always disassemble the machine code extension file provided with Hermes in order to see how Tony Firschmann has used the MT.IPCOM command. Having done this, you can (after some work) find out how to access Hermes from machine code (the machine code

extensions utilise all of the additional Hermes commands), although the way in which this switches on Hermes is a little obscure.

The original Microdrive Reduced Sensitivity command only allows one parameter, however, the extension file seems to pass two parameters, the second of which would appear to be used to call the MT.IPCOM Test command at the same time. Still, machine code programmers need not worry why this is done in this way, as they need only copy the code into their own programs.

This could have been made a lot easier by the incorporation of either a fully commented source code for the extension file, or even an example in the manual itself. I have forwarded to Tony a copy of the list of commands which I produced, and maybe this can be incorporated into a later version of the manual.

Improvements

The other improvements which are noticeable when Hermes is installed are improvements to the BEEP command - no longer does the pitch of the note dictate how long it will be sounded for, nor do the fuzzy and random parameters alter the actual pitch of the sound. Many people will have come across the key combination Ctrl-Alt-7, which was originally used to turn on some test equipment which only Sinclair owned (and to reset the 8409), and crashes the computer without this equipment. This keystroke now produces CHR\$(255) plus CHR\$(151), which as the QL Manual suggests. Instead, the keys Ctrl-Alt-Esc-7 can now be used to reset Hermes (which also crashes the computer), which is somewhat more difficult to press by accident.

There are one or two problems which could have been addressed by Hermes but have not been yet. No extra baud rates have been added to the QL's repertoire (the current eight are still supported), although the manual suggests

that these may be added in the future. KEYROW still suffers from the same problem that when three keys are pressed to form three corners of a rectangle in the keyboard matrix shown in the QL Manual, the key which appears at the fourth corner in the grid also seems to have been pressed. Also, according to the QL Manual, the Shift, Alt and Ctrl keys should be an exception to this rule however, on my old version of the 8049 they aren't, nor are they on Hermes. Maybe this is just another example of an inaccuracy in the QL Manual rather than a design flaw.

Despite the Hermes manual being a little too technical for some users (hopefully this will be improved as time goes by), Hermes manages to achieve all that it sets out to do, and I recommend it to anyone who has had problems with the 8049 in the past, as well as those who want the extra facilities afforded by it.

Living with Hermes

INFORMATION

Product: Hermes replacement co-processor (V2.18)

Supplier: TF Services, 12 Bouverie Place, London W2 1RB

Price: £25 (£24 to Europe, £27 outside Europe) (state whether software on mdv or disk)

DIY Toolkit

Simon Goodwin explains screen-scanning and task generation in the second part of HPDUMP.

This article explains how HPDUMP converts a QL screen into a grey scale image suitable for any printer that supports Hewlett Packard Printer Control Language. It also presents a multi-tasking version, HPDUMP_TASK, which can replace Psion's GPRINT_PRT in some applications.

Example screen

The example screen comes from QL Xchange, via Dr. Bill Fuggle's LaserJet II printer, and shows that the dump routine does not distort the shape of circles, even though there is no simple correspondence between QL pixels and dots on the page printer.

Listing One is the second half of last month's assembly code, and contains the routines that read the QL screen and send data to the printer. Before the screen can be printed, HPDUMP must tell the printer the graphics resolution and line width. The Mode4 screen uses 512 pixels per row.

The string labelled SETUP150, near the end of Listing One, selects lines of 1024 dots at 150 dots per inch, giving an image a little under seven inches wide if we allocate two dots per pixel. If the value t150R is replaced with t300R, you get the maximum Deskjet resolution of 300 dots per inch, and the image is halved in width and height, a little over two inches tall.

Mode4 screens

The Mode4 screen memory uses alternate bytes to store red and green components of the picture. Each pair of eight bits in adjacent bytes controls the colour of a single pixel. You get eight bits of green data, then eight red bits, and so on.

If HPDUMP sent these bits to

the printer in the same order, red and green areas would appear as offset stripes, and the screen image would consist of 128 stripes with black and white repeated.

The screen needs to be unscrambled to suit the printer, with grey shades for red and green pixels and solid areas for black and white. The grey shades use patterns of dots, like QL stipples - try typing some SuperBasic commands after PAPER #0,4,0 or PAPER #0,0,2,0 to see the effect I mean.

As the significance of each screen bit repeats for every word, we could translate a group of eight pixels by looking it up in a table of 65536 entries, one for each possible word value.

Each screen word might be represented as a row of 16 or more bits representing printer dots, in the right order. This would be very fast compared with routines that work a pixel at a time, like RECOL, but it needs lots of memory: 128K if we use two printer dots per pixel for grey shades, or even more with wider patterns for each pixel.

HPDUMP uses a table of just 512 bytes. It converts each green and red byte in turn then merges them together with OR instructions. This compromise is still quite fast, but less greedy for memory than the 128K table. The HPDUMP table holds 256 words, with a value 0 to 255 encoded in the even bits, and zero in the others.

Each screen pixel corresponds to a printed pattern two dots wide and two or three dots high. This ratio makes the printout the same shape as the display. The code from DO_LINE to SKIP_LINE in Listing one sets all six dots for a white pixel, three for green or two for red.

The BTST #0,D6 instruction before the third line of the pattern is generated means that the third line is skipped for every

alternate pixel row. D6 counts the lines, so bit zero changes for each row of pixels. If you remove that line, and the following BNES SKIP_LINE, the printout will be taller than the display but grey tones will be more consistent, as every row of pixels gets treated the same way.

Expansion table

The expansion table values could be preset and stored in the file, but this would mean lots of tedious typing for readers, so I decided to generate it algorithmically. The code was sure to be much more quickly entered than the 512 bytes of data, and would occupy less file space.

The disadvantage is that both code and data occupy memory while HPDUMP is running, but this is not much trouble as the routine already needs a buffer to store data en route to the printer, and the table can fit alongside that allocation. The code uses 512 bytes for the table, and 128 bytes to buffer a single print line.

If you invoke HPDUMP_TASK with CALL, or use HPDUMP_CODE, 640 bytes of temporary space are allocated on the common heap while it runs, and released afterwards. The task version has no need to allocate heap space, as it has 660 bytes of private data-space when loaded.

The extra 20 bytes allow some breathing space for the task parameters, if any. If the parameter string is long it will be overwritten by the table, but that does not matter as HPDUMP will have read it and opened the corresponding file by that time.

The first version of the table-generating code was very simple - it just counted from 0 to

255, and tested each bit in the count byte to determine whether or not an even bit should be set in the corresponding word. BTST and BSET instructions use four bytes each, so the resultant code seemed rather long and slow.

Discussions with Phil Spink led to the current version, which is less than half the size and three times as fast. It uses sixteen data bytes and 38 bytes of code to expand that into the full 256 words. The code may appear complicated, with its double indices and nested loops, but in fact each step is quite straight-forward, combining two bytes from the table into one data word.

Register A3 points at the expanded table, while A1 points at the first of sixteen bytes of 'nibble' data. A nibble, or nybble, is a fraction of a byte. In this case the nibbles count from 0 to 15, using only the even bits of a byte, leaving the odd bits unset. By convention bits are numbered from zero, the least significant, upwards, so bit 7 is the most significant bit of a byte.

D0 selects the most significant byte value from the table, while D1 selects the least significant, and D2 points at locations for the resultant word. DBRA loops count downwards, so the table is filled from the end backwards, but the order of calculation makes no difference to the result.

Code vs data

It's possible to imagine ways to build the table faster or from less data. For instance you could build the sixteen byte table from four nibbles, and then use much the same code to expand the bytes to 256 words,

but in practice the extra code outweighs the saving in data. For top speed, at the expense of space, there's no substitute for pre-computed data, and the more the better.

The routines DO_LINE, DO_LINE2 and DO_LINE3 vary to give patterns for grey rather

line at a time than to write the bytes two by two as they are generated by scanning the screen.

Each line of dots is prefixed by the text labelled NEWLINE, which tells the printer to expect 128 bytes. Notice that the number is written in decimal,

HPDUMP_TASK, a multi-tasking version of the HPDUMP command. Type this in and RUN it to generate the HPDUMP_TASK file on your chosen device.

Notice that line 160 has been changed from the usual SBYTES to SEXEC, and an extra parameter ,660 appears at the end of that command. These changes reserve the data-space needed by the task. EXEC reports 'bad parameter' if you try to load a file without such data-space. The rest of the first part of listing Two is the same as usual, so you need not re-type it if you have an earlier Toolkit loader on file.

The hex data for the task follows, from line 580 to 920. The length is the same as for last month's HPDUMP_CODE, but this is a coincidence - the data bytes are different.

HPDUMP stars in the latest DIY Toolkit volume, which explores the details of HP Print Control Language. Volume K includes the task, CALL and command versions of

HPDUMP, full source code, configuration data for Quill and Xchange, dozens of SuperBasic extensions to select printer fonts and styles and a TRA code translator for special characters, which suits all Qdos variants except AH and JM.

There are now 24 volumes of DIY Toolkits available on disks and microdrives. Prices have been cut to a flat rate of £3 each on 3.5 or 5.25 inch Qdos disks, or £4 volume on cartridge. Orders are despatched by first class or airmail post, included in the price.

Please send sterling cheques, postal or money orders - not

credit cards. If you order two or more volumes you will receive a laser-printed leaflet of documentation for each, at no extra charge. To obtain volumes, or further information, write to Dr. Bill Fuggle at DIY Toolkit, 86 Lordswood Road, Harborne, Birmingham B17 9BY, UK

Using HPDUMP

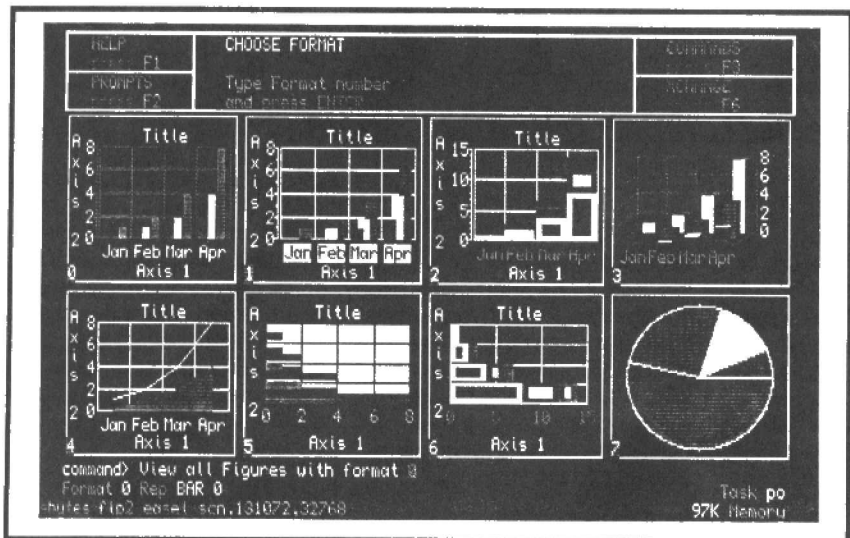
HPDUMP_TASK can run like any other task, from the standard EXEC or EXEC.W commands. If you use EXEC.W the SuperBasic interpreter waits until the printout is complete before re-starting. EXEC FLP1-HPDUMP_TASK lets you carry on using the machine, but if you change the screen while the task runs the printer will show a mixture of the old and new screens, and even if you type nothing the waiting cursor may end up on the printout.

Minerva users have no such problem as they can swap to the other screen once printing has started, and carry on using that display while HPDUMP prints the other one. The task automatically detects which screen is in use when it is invoked, and continues to print from that one even if the user selects the other.

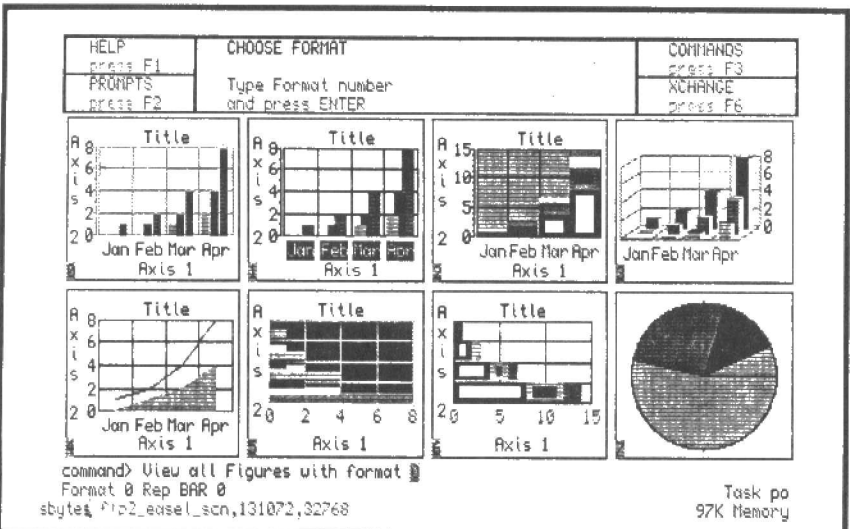
If you use a Sinclair rom, Thor or other emulator, and don't want to wait for the printer, you need to re-direct the output of the task to a buffer which can be printed later. There are lots of ways to do this, often relying on extended versions of the EXEC command which let you pass parameters to the task.

Some systems, like the Thor and SuperQboard, can insert a buffer on the way to the printer, where data is stored temporarily until the printer is ready for it. These work well with HPDUMP, but you need to reserve about 100K for the information used by a single dump. In this case EXEC.W HPDUMP_TASK will complete in about a second, leaving the screen ready for other uses, but the printer will carry on for a minute or more, using data from the buffer.

If you have QJump's RAMPRT driver installed, and a Toolkit that allows EXEC parameters, you can direct the dump to the buffered print device PRT. This works much like the buffered SER or PAR device, except that you need to pass the name PRT to over-ride the default of SER.



Grey scales on a black and white background



than simple stripes. DO_LINE2 only sets the odd bits if both green and red bits are set on the screen.

Each resultant word is exclusive ORed with the value in D5 to exchange black and white dots unless the ! parameter was used. If all bits of D5 are set, this reverses the brightness just before the data is stored in the buffer.

SEND_LINE sends 128 data bytes at a time to the printer using SEND_STRING twice. First it sends the control sequence, then the data bytes from the buffer addressed by A5. It is much more efficient to send a

rather than encoded into a byte or word as it would be for an Epson printer. This makes PCL escape sequences relatively long but easy to read as all the characters are Ascii codes. The last letter of an escape sequence is capitalised, to mark the end.

Once the DO_SCREEN loop has finished, HPDUMP checks the value of D7 to determine whether it should close the channel or release the memory buffer.

Listing two

Listing Two is a hex loader for

This is the SuperToolkit 2 command:

```
EW HPDUMP_TASK;"PRT"
```

Use QW instead of EW if using the Qliberator toolkit. The Turbo Toolkit equivalent is:

```
EXECUTE_A  
HPDUMP_TASK;"PRT"
```

Add an exclamation mark at the end of the name if you want a true grey scale, rather than the default which saves ink by swapping black and white in the printout.

If you always have QJump's RAMPRT, or its equivalent, loaded you can patch the task to replace the device name SER with PRT, and call it with a normal EXEC command. If you don't want to re-assemble the source, the easiest way to patch the task is with The Editor or Spy. Adam Denning's FEDIT will do the job, but scrambles the task's data-space so you must reset it to 660 bytes after editing with LBYTES and SEXEC, DATASPACE_TASK from Supercharge or Turbo, or SetHEAD, from DIY Toolkit Volume F.

If you intend to print the same image repeatedly it is worth re-directing the output of HPDUMP_TASK to a file, and copying that to the printer as required. The Toolkit 2 SPL command is useful here, as it can be directed to any device and uses a background task to copy the information while you get on with something else. Alternatively, use a compiled COPY command.

On call

Although the file is named HPDUMP_TASK, you can also invoke it with a CALL command, rather like Psion's GPRINT_PRT. Some commercial programs generate an Epson printer dump by loading GPRINT_PRT and calling the start of its code, as explained in this month's SuperBasic in Action column.

Programs that do this, such as 3D Terrain, work just as well with an HP PCL printer if you replace the supplied GPRINT file with a renamed copy of HPDUMP_TASK. The code can tell whether it was invoked by EXEC or CALL, and checks its parameters appropriately.

Unfortunately HPDUMP_TASK cannot be used directly as a

```
* QL World DIY Toolkit, HPDUMP for Qdos screens, Part II
* Version 2.18, Copyright Simon N Goodwin March/April 1993
*
* Continued from Listing 1, QL World 1993 issue 5, page 28
*
* Send initialisation sequence
*
        moveq    #-1,d3          Infinite timeout
        lea.l    setup150,a1     Point at the string
        move.w    (a1)+,d2       Byte count
        bsr

*
        movea.l   a2,a3          Copy table pointer
        lea.l     nibbles,a1     Point at nibble data
*
* A5 -> line buffer, A3 -> pixel table, A1 -> Nibble table
*
high_loop  moveq    #15,d0       Outer loop counter
low_loop   moveq    #15,d1       Inner loop counter
           move.w    d0,d2       Form target index in D2
           lsl.w     #4,d2
           add.w     d1,d2       D2 := D0 * 16 + D1
           add.w     d2,d2       D2 is a word index
           move.b    0(a1,d0.w),d4
           lsl.w     #8,d4       D0 points at high byte
           add.b     0(a1,d1.w),d4 D1 points at low byte
           move.w     d4,0(a3,d2.w) Store combined word
           dbra      d1,low_loop
           dbra      d0,high_loop

*
* Print out the screen addressed by A4
*
do_screen  move.w    #255,d6     Line count
           moveq     #63,d0     Do 64 words
           movea.l   a5,a3       Build line at A3
do_line    moveq     #0,d1       Green byte
           move.b    (a4)+,d1     Make word index
           add.w     d1,d1
           move.w     0(a2,d1.w),d2
           add.w     d2,d2       Move odd to even bits
           moveq     #0,d1
           move.b    (a4)+,d1     Red byte
           add.w     d1,d1       Make word index
           or.w      0(a2,d1.w),d2 Mix red & green data
           eor.w     d5,d2       Apply INVERSE/TRUE mask
           move.w     d2,(a3)+    Buffer one word
           dbra      d0,do_line
           bsr.s     send_line

*
           lea.l     -128(a4),a4  Go back again
           moveq     #63,d0     Do 64 words
           movea.l   a5,a3       Build line at A3

do_line2   moveq     #0,d1       Green byte
           move.b    (a4),d1     Make word index
           add.w     d1,d1       Get the odd bits
           move.w     0(a2,d1.w),d2
           moveq     #0,d1
           move.b    (a4)+,d1     Green again
           and.b     (a4)+,d1     Green AND red?
           add.w     d1,d1       Make word index
           move.w     0(a2,d1.w),d1
           add.w     d1,d1       Get the other bits
           or.w      d1,d2       Make them even
           eor.w     d5,d2       Combine odd and even
           move.w     d2,(a3)+
           dbra      d0,do_line2
           bsr.s     send_line
           bne.s     skip_line   Skip one line in six

*
* Send the final line of dots for this QL pixel line
*
           lea.l     -128(a4),a4  Go back again
           moveq     #63,d0     Do 64 words
           movea.l   a5,a3       Build line at A3
do_line3   moveq     #0,d1       Green byte
           move.b    (a4)+,d1     Make word index
           add.w     d1,d1       Get the odd bits
           move.w     0(a2,d1.w),d2
           moveq     #0,d1
           move.b    (a4)+,d1     Red byte
           add.w     d1,d1       Make word index
           move.w     0(a2,d1.w),d1
           add.w     d1,d1       Get the other bits
           or.w      d1,d2       Make them even
           eor.w     d5,d2       Combine odd and even
           move.w     d2,(a3)+
           dbra      d0,do_line3
           bsr.s     send_line

*
skip_line  dbra      d6,do_screen
           moveq     #0,d4

*
close_out  tst.l     d7           Check EXEC/CALL flag
           bmi.s     d4_error    No CLOSE if task parameter
           bne.s     tidy_up     No RECHP for tasks
           exg       a5,a0
           moveq     #25,d0
           trap      #1
           exg       a5,a0
           moveq     #2,d0
           trap      #2
           move.l     d4,d0       Return error code
           rts
bad_exit
```

```

*
* Write the line of pixels at A5 to channel A0
*
send_line    lea.l    newline,a1
              move.w  (a1)+,d2          Fetch length
              bsr.s   send_string      Initialise line
              bne.s   fall_out
              movea.l  a5,a1            Point at the bytes
              move.w   #128,d2          Byte count
send_check   bsr.s   send_string
              beq.s   exit_ok
fall_out     addq.l   #4,a7             Discard return address
              bra.s   close_out
*
send_string  moveq    #7,d0             IO.SSTRG
              trap    #3
              move.l   d0,d4           Set Z flag if it worked
exit_ok      rts
*
* Table of values 0 to 15 in even bits, with odd bits zero
*
nibbles      dc.b    0,1,4,5,16,17,20,21,64,65,68,69,80,81,84,85
*
* HP PCL escape sequences, at the end for easy patching
*
printname    dc.w     4
              dc.b     'SER1'
newline      dc.w     7
              dc.b     27,'*b128W'      128 bytes per line
setup150     dc.w     15
              dc.b     27,'*t150R'      150/300 DPI density
              dc.b     27,'*r1024S'     1024 pixels per line
*
              ifne     extension
*
define       dc.w     1                One procedure
              dc.w     hpdump-*
              dc.b     6,'HPDUMP'
              dc.w     0,0,0           That's all
*
              endc     extension
*
              end

```

replacement for GPRINT_PRT, as Psion include a collection of undocumented data values at the start of their routines and these are absent from HPDUMP, so Easel rejects the file and Xchange refuses to install it. I hope to work out the format in due course, and update volume K appropriately, but in the meantime you can print Easel screens by saving them to disk or microdrive, reloading them with LBYTES, and printing them from SuperBasic, like this:

```

LBYTES
FLP1_PSION_SCREEN,131072
EXEC_W
FLP1_HPDDUMP_TASK

```

I'd be most interested to hear from anyone who has details of the Psion _PRT header format, and as always I welcome comments and suggestions from DIY Toolkit readers.

Next month I'll be back with an in-depth report on a new version of Psion's Xchange suite

DIY TOOLKIT HPDUMP_TASK loader - LISTING 2

```

100 REMark Sinclair QL World HEX LOADER v 3b
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
130 CLS: RESTORE : READ space: start=RESPR(space)
140 PRINT "Loading Hex..." : HEX_LOAD start
150 INPUT "Save to file...";f$
160 SEXEC f$,start,byte,660 : STOP
170 :
180 DEFINE FuNction DECIMAL(x)
190 RETURN CODE(h$(x))-48-7*(h$(x)>"9")
200 END DEFINE DECIMAL
210 :
220 DEFINE PROCEDURE HEX_LOAD(start)
230 byte = 0 : checksum = 0
240 REPEAT load_hex_digits
250   READ h$
260   IF h$="" : EXIT load_hex_digits
270   IF LEN(h$) MOD 2
280     PRINT"Odd number of hex digits in: ";h$
290     STOP
300   END IF
310   FOR b = 1 TO LEN(h$) STEP 2
320     hb = DECIMAL(h$(b)) : lb = DECIMAL(h$(b+1))
330     IF hb<0 OR hb>15 OR lb<0 OR lb>15
340       PRINT"Illegal hex digit in: ";h$ : STOP
350     END IF
360     POKE start+byte,16*hb+lb
370     checksum = checksum + 16*hb + lb
380     byte = byte + 1
390   END FOR b
400 END REPEAT load_hex_digits
410 READ check
420 IF check <> checksum
430   PRINT "Checksum incorrect. Recheck data.":STOP
440 END IF
450 PRINT "Checksum correct, data entered at: ";start
460 END DEFINE HEX_LOAD
470 :

```

```

580 REMark Space requirements for the machine code
590 DATA 484
600 :
610 REMark Machine code data
620 DATA "600E322E31384AFB","0006485044554D50"
630 DATA "41FAFFEEBDC8665A","7E01301F670A76F1"
640 DATA "534066487EFF205F","7A0030176F160C37"
650 DATA "0021000166067AFF","535767084A876B4E"
660 DATA "204F60084A876B1A","41FA017872FF7602"
670 DATA "70014E4226006614","7001D05F08800000"
680 DATA "DFC04BF6C0002C08","613A260072FF7005"
690 DATA "4E417AFF4A816602","7A0041FA01467E00"
700 DATA "72FF760070014E42","4A80660001042C08"
710 DATA "223C0000028074FF","70184E4128006600"
720 DATA "00EA2A4845ED0080","287C000280007000"
730 DATA "4E41102800346B04","49EC8000204676FF"
740 DATA "43FA011034196100","00DA264A43FA00E4"
750 DATA "700F720F3400E94A","D441D44218310000"
760 DATA "E14CD83110003784","200051C9FFE851C8"
770 DATA "FFE23C3C00FF703F","264D7200121CD241"
780 DATA "34321000D4427200","121CD24184721000"
790 DATA "BB4236C251C8FFE4","617849ECFF80703F"
800 DATA "264D72001214D241","343210007200121C"
810 DATA "C21CD24132321000","D2418441BB4236C2"
820 DATA "51C8FFE0614C0806","0000662A49ECFF80"
830 DATA "703F264D7200121C","D241343210007200"
840 DATA "121CD24132321000","D2418441BB4236C2"
850 DATA "51C8FFE2611C51CE","FF7E78004A876B0E"
860 DATA "6608C14D70194E41","C14D70024E422004"
870 DATA "4E7543FA00343419","6110660A224D343C"
880 DATA "00806106670A588F","60D270074E432800"
890 DATA "4E75000104051011","1415404144455051"
900 DATA "5455000453455231","00071B2A62313238"
910 DATA "5700000F1B2A7431","3530521B2A723130"
920 DATA "32345300","*",39775

```

MicroEmacs 3.11

Hilary Snaden

Following the review of version 3.9p of the public domain text editor *MicroEmacs* in *QL World*, August 1992, version 3.11 has been released. The new version is based on an updated source code together with a rewrite of QL-specific code by Richard Kettlewell.

Included on the disks are a number of utilities written in the MicroEmacs command language, together with a number of machine code programs for file manipulation which may be invoked from Basic or from MicroEmacs itself since the commands which invoke outside programs now work. Environment variables are also supported, if the necessary extensions to superbasic have been installed.

The program now copies the DATA_USE and PROG_USE devices for its own use if Toolkit 2 is present, and if started from a command line wildcard filenames are accepted so that, for example, it can be set to read a series of files. MicroEmacs can also now complete filenames in the same way as buffer and procedure names.

The way that function keys are interpreted is now more in line with established QL conventions, and the help facility has been substantially expanded. Paper and ink colour can be changed properly, and the QL's clock can be read as a MicroEmacs variable and added to documents: this is used by one of the new utilities, which produces a three-month calendar, centered on the current date, which can be incorporated into documents.

Commands such as list-buffers and describe-bindings now use a volatile window which disappears, restoring the original screen, on the next keypress. Screen handling is noticeably faster thanks to a major rewrite of the display code. MicroEmacs is fully compatible with Minerva.

One potential problem remains: memory grabbed by MicroEmacs for storing files being edited is not released until the program is exited.

MicroEmacs 3.11 as source code and a ready-to-run program, together with documentation, is available from **public domain libraries**.

QXL'S Brum Debut

Simon Goodwin

The new QXL from Miracle Systems was unveiled at a packed meeting of QL enthusiasts at Birmingham's Holloway pub, home of the West Midlands Quanta sub-group. The QXL is a fast QL emulator board for PC compatibles, based around Motorola's 32 bit EC 68040 processor.

The QXL was shown alongside the Gold Card, and succeeded in making even that look slow by comparison. Stuart Honeyball started by running QL Quill on the PC-based QXL, copying lines repeatedly to create a file of several thousand words in a few seconds.

Even programs that write directly to QL memory work on the QXL, as the 68040 copies the 32K QL screen to the PC display adapter. Normal QL displays appear in the middle of a larger VGA screen, and programs that do not write directly to the QL display memory can use the extra space for text or graphics. The current version is limited to Mode4, but Miracle plan to add support for Mode8 in due course.

The QXL has its own memory and only needs the PC as a peripheral controller. It starts up from two disks, one in PC and one in Qdos format. Once the system is loaded users can return to the MS-DOS prompt, issue DOS commands, and re-start the QXL at the point they left off.

The SMS4 operating software was still in prototype form at the sub-group meeting, limiting testing. There was no SuperBasic interpreter, so Toolkits could not be loaded and it was only possible to run stand-alone QL tasks.

The only PC device supported was the floppy drive, at a leisurely speed compared with a QL drive, but programmer Tony Tebby plans to fine-tune the speed and add support for SER, PAR, NET and WIN devices over the summer. Miracle promise free updates when the software changes.

The QXL was happy to run the PD version of QL *Xchange* from the sub-group's library, as well as *Quick Mandelbrot III* supplied by Rich Mellor. This revealed one apparent glitch, showing a co-ordinate ten times the correct value, and Miracle and Mellor are collaborating to trace the source. The display was correct, and generated two to three times faster than by the same code on a Gold Card.

The QXL costs between £295 and £495 depending on memory configuration (one to eight megabytes) and requires a PC XT or compatible with at least 512K ram, EGA or better display, and one free slot. It will work on a basic 8-bit PC but benefits from the extra signals of a 16-bit ISA slot. The development version expects the PC to have AMI BIOS chips, but should be largely compatible with the Phoenix alternative.

New Discoveries

Simon Goodwin

Prolific C programmer Dave Walker has extended his *DiscOver* and *Multi-DiscOver* utilities to read and write HD and ED disks in PC format, on any QL with a Gold Card and a suitable drive. The new version will also format 1.44 megabyte HD disks in PC or QL format.

The PC's little-used 288 megabyte format is not available, as the Gold Card uses larger sectors to put 3.2 megabytes on each ED disk, but the upgraded *DiscOver* can read, write and erase files on ED disks formatted by a PC.

Multi-DiscOver version 3.01 can also access files in the MGT disk formats used on many Spectrum and SAM computers. Again it is possible to read, write and erase files with any double-density drive, but the program cannot actually format disks in MGT's 800K ten-sector format.

A new format specification file allows easy access to 706K Amstrad disks. These come from later Amstrad PCW machines and the Spectrum Plus Three with external drive B: fitted. The same format is also used by ProDOS, the SAM CP/M system, so it is now possible to convert files from those sources to suit Qdos CP/M emulators.

Freely-distributable demonstration versions of *DiscOver*, *Multi-DiscOver* and *Textidy* are now available from QL Bulletin boards and PD suppliers. The demo versions come with complete documentation in *Quill* files on the disk, and are fully functional except that it is not possible to write files of more than 512 bytes. You can read, erase and format disks, and check that the programs do what you want.

The demonstration tasks and full programs are available from **Dilwyn Jones Computing, tel. 248 354023**. *Discover*, for QL, PC and ST formats, costs £20, while *Multi-DiscOver* adds Spectrum, CP/M, Unix and BBC file support, for an extra £10. The file conversion utility *Textidy* is priced at £15. These programs require a QL with at least 256K of memory and suitable drives.

QLScene

SINCLAIR QL WORLD

JUNE
37

PSION *Solutions*

Howard Clase with two more routines for Quill users.

Continuing

the theme of SuperBasic programs to ease the lives of Quill users here are two more ideas, one inspired by a correspondent in a North American QL newsletter, and the other in response to no less person than QL World's own editor.

Make Your Own Printer Driver.

One of the most frequent requests from Quill users is for a way to increase the ten "translates" in the Psion printer driver which enable you to send special instructions to the printer, such as to turn on italics, or to get the printer to print characters from the second character set correctly. One way around this is to use the QL's second character set as control codes with your printer set to receive 7-bit Ascii (for many printers this is the default - see Psion Solutions, April 1991). But what if you want to print a large number of foreign characters from the second set, text containing both French and German for example, which would require at least 12 additional characters? The answer is, of course, to do it in SuperBasic, using a supplementary printer driver.

A printer driver has to check each character before it is printed to see whether it should be sent to the printer as it is, or translated into something else, like the codes for switching on superscript, or to substitute the

character for the #. It isn't difficult to write a program to do this, but since we cannot run a SuperBasic program from within Quill it has to be a two step process. For once the slow speed of interpreted Basic isn't a problem - it has no difficulty keeping up with most printers!

A skeletal program of this kind is given in Listing one. It is designed to work on a ".lis" file - the kind that is generated from Quill when you type in a file name instead of pressing Enter when the words "to printer" appear at the bottom of the screen during the printing sequence. (As you start to type, the word "printer" will vanish so that you can enter a file name of up to eight characters and, unless you add a different device and/or three letter extension of your own, the file will appear on your default device with the ".lis" extension.) This file is exactly the set of bytes that would have been sent to the printer via your current printer.dat driver, and will already contain all its printer control codes and translates. For the Psion suite, listing five is a supplementary printer driver; it adds to Quill's internal printer driver. You need to bear this in mind when writing your program. But, although I started out with the idea of supplementing the Psion printer driver the program is quite general and will work on any listing that is basically text, in particular, SuperBasic listings that include foreign characters. Listing five was used to print

itself on my old Epson FX85; I couldn't just LIST it to my printer since it contains second set characters in the REM statements.

The Program

Because there are almost as many models of printer as there are QL users there is no point in my providing a complete program that would be useless to many readers, so listing five is just a skeleton containing examples of the kind of thing you can do. I'm afraid it's up to you to struggle with your printer manual and flesh out the assignments exemplified in lines 160 to 200 to meet your requirements. You can extend this collection of definitions by using the RENUMBER command to make more lines. All the codes in the listing are Epson ones, and the examples do work on a genuine Epson. You may have to change some or all of them for other brands; a claim of "Epson compatibility" is no guarantee of an 100% match in my experience. One thing you will need to pay attention to is the printer channel definition in line 145; most people use ser1 here.

The second character set occupies the 64 Ascii values from 128 to 191. A "look_up" array, new\$, to hold 64 strings of up to 10 characters is set up in line 155 and initially filled with spaces; change the second parameter if you want to use longer strings. NB: the DIM statement only reserves even numbers; odd requests are increased by one. Add 127 to the array index to get the corresponding Ascii code such

as new\$(1) corresponds to CHR\$(128). The special characters can be called up from your printer in two ways: by sending the codes to select a foreign character set containing the desired character (lines 160, 170, 195) if it is available, or by a bit of creative overprinting e.g. the division sign by a dash and a colon (lines 165, 185, 200 - bs\$, CHR\$(8), is backspace.) If you want to leave a character in the 128 to 191 range unchanged you must put it back in again (175, 180). I chose the space as default since not all of the characters can be generated by these methods and it gives the opportunity of insertion by hand. Note that the assignments must be characters or strings: defined as CHR\$(), one or more characters in quotes, or a string variable; and they must be combined with &s. These are assigned to the array as required to replace the spaces.

Mnemonics

I hate having to type the same thing repeatedly, so I have defined a set of string variables (105 - 130) to call up the various language codes in mnemonic form (Fr\$ = French, etc). The default on my printers is US mode (there isn't a Canadian one!), and this has to be restored after each foray into foreign parts; to save typing again, this is automatically tacked on to every member of the array which is more than three characters long at line 210. (If it's shorter than this it cannot contain a nationality change.) If necessary, change this to match your required default character set; for instance, use Sw\$ if you

normally type in Swedish.

As written, the program only checks the characters from the second character set with Ascii values from 128 to 191 and the "" sign (CHR\$(96)), but this can easily be altered (lines 325 to 240). (The code at line 335 is a one off to deal with the ""). The normal characters and printer control codes all fall below 128, and are passed on unchanged (325). If your printer_dat processing sends out any characters in the 128 to 191 range then you will have to watch out for conflicts, but this shouldn't be a serious problem.

If you have Toolkit 2 you can save also quite a bit of typing if you set up 'new\$(0)=\$&CHR\$(0)' and 'new\$(0)=""&bs\$&"' on a couple of ALTKEYs, and insert the missing numbers and characters as required. I find that if I am generating a composite character using backspace, I have to use unidirectional printing to get a tidy looking result (line 150). Try it without this on your printer too, since it saves time if you can print bi-directionally. You'll have to experiment here, as what works in one print style may not look right in another. If you are not sure about the keyings of some of the characters refer to "character set and keys" pages of the "Concepts" section of the QL manual.

Some printers which accept 8 bit Ascii, particularly those with IBM character sets, do in fact have most of the required characters, but with different values from the QL ones. In this case, since all that is needed is to substitute a single code in each case, it is probably simpler to put the new values into DATA statements and READ them into the array from these.

Exporting Paragraphs

In Psion Solutions for November '92 your editor commented that it really simplifies the desktop publishing process if Quill documents are exported to Ascii without any control codes or other formatting except for paragraph separations and an EOL (end-of-line - I presume the MS-DOS form "CR,LF" is required) at the end of each paragraph. Here are a couple of ways it can be done.

My first thought was that it would have to be done in two steps, using a SuperBasic utility on an ascii _lis file, but when I had worked this out I realised

that it is after all possible to do it directly from within Quill. Both methods require a bit of "de-formatting" of the Quill document and specially simple printer_dat files. I think the latter is the most convenient way, so I'll give it first and in more detail.

Printer dat

The special printer_dat driver has to be created using install_bas. Select the default driver, because it is already pretty simple, and copy it by pressing F1. Change the name to something else, set the lines per page to 0, and, most important, alter the end of line code so that it just inserts a space. To do this press the Right arrow to remove the old entry and then type a single apostrophe, then a space, and then hit Enter. Since the entry is a space it won't appear to be there, but the word "NONE" won't appear either. No EOL codes will appear at the end of each Quill line, and the last word of a line will be properly separated by a space from the first word of the next line. Leave preamble and all the type style entries as "NONE". It's a good idea to put in "CR,LF" as postamble to make quite sure there's an EOL at the end of the file.

So far so good, but there must be an EOL at the end of each paragraph; since the driver doesn't allow you to define an EOP (end-of-paragraph) this has to be done using a translate. I chose the greek mu character, since that's what you get when you press Ctrl-Shift-P, but you can use anything you like that you can easily remember the keying for and which isn't likely to appear in the text. Move to translate 1, press Right arrow and then type "<greek mu>,CR,LF,CR,LF" (omit the double quotes, key in <greek mu> as above, and insert enough CR,LFs to give a two-line separation between paragraphs. This means that every time the driver meets a mu it inserts the MS-DOS EOL codes to close the previous paragraph, and leaves an extra line (with returns) before the next.

Entries relating to the serial port and page length etc. are irrelevant since you aren't going to be using this driver with a printer - if you do you may get rather odd results! If you are producing hardcopy to go with a disk file (the editor tells me) then the format doesn't matter too much as long as the words are right. The disk file format is the

important one.

Press F5 to install the driver as printer_dat on your default device, and then copy it to a file with a meaningful name so that you don't have to go through this rigmarole every time you want to carry out the operation.

De-formatting

The printer driver creates the left margin by sending the appropriate number of spaces to the printer at the beginning of each line, and the last thing your editor wants is a set of ten spaces inserted at regular intervals throughout the text! Nor does she want it to be punctuated by footers or headers every so often, so the Quill document has to be "de-formatted" before it is exported. Save it before you do this to

top of your document and set justification F3-J to "left" to remove surplus spaces, and finally move both left and indent margins to the left hand edge of the screen F3-M. If there are no changes to justification or margins elsewhere in the document these should run right to the bottom; it's just as well to pop down there to make sure. If not, you'll have to go down the document paragraph by paragraph using Shift-Down-arrow, and reset the justification and margins wherever necessary.

Now for the most tedious bit in this system, you have to put a mu at the beginning of every paragraph. This isn't too bad when you realise that pressing Shift-Down-arrow moves you straight to the right place each time. If you have TK2 you can

```
10 nm$="DIY_Printer_driver"
15 REMark (pd) hjc 1992.07.13 ver 1.0
20 REMark ~~~~~Main
25 Setup: FPrint: Finish: STOP
30 REMark ~~~~~Setup
100 DEFine PROCEDURE Setup
105 Pr$=CHR$(27)&"R": bs$=CHR$(8)
110 US$=Pr$&CHR$(0): Fr$=Pr$&CHR$(1)
115 Ge$=Pr$&CHR$(2): UK$=Pr$&CHR$(3)
120 Sw$=Pr$&CHR$(5)
125 It$=Pr$&CHR$(6): Sp$=Pr$&CHR$(7)
130 Ja$=Pr$&CHR$(8): No$=Pr$&CHR$(9)
135 CLS: INPUT "Device and file name to print: ",n$
140 c%=3: OPEN_IN#c%,n$: REMark Unused channel
145 OPEN#c%+1,ser2hc: REMark Your printer channel
150 PRINT#c%+1,CHR$(27)&"U1": REMark one way
155 DIM new$(64,9): FOR i=1 TO 64: new$(i)=" "
160 new$(1)=Ge$&CHR$(123): REMark a
165 new$(2)="a"&bs$&"": REMark a
170 new$(3)=Sw$&"": REMark a
175 new$(11)=CHR$(136): REMark unchanged
180 new$(15)=CHR$(142): REMark unchanged
185 new$(16)="e"&bs$&Sp$&CHR$(123): REMark e
190 new$(28)="u"&bs$&"^": REMark u
195 new$(32)="f": REMark f
200 new$(60)="-"&bs$&"": REMark -
205 FOR i=1 TO 64
210 IF LEN(new$(i))>3: new$(i)=new$(i)&US$
215 END FOR i: END DEFine: REMark ~~~~~FPrint
300 DEFine PROCEDURE FPrint
305 LOCAL a,loop: REPEAT loop
310 IF EOF(#c%): EXIT loop
315 a=CODE(INKEY$(#c%,-1))
320 SELECT ON a
325 = 1 TO 95, 97 TO 127, 192 TO 255
330 PRINT#c%+1,CHR$(a);
335 = 96: PRINT#c%+1,UK$&"#&US$; REMark &
340 = 128 TO 191: PRINT#c%+1,new$(a-127);
345 END SELECT: END REPEAT loop
350 END DEFine: REMark ~~~~~Finish
400 DEFine PROCEDURE Finish
405 CLOSE#c%: CLOSE#c%+1
410 END DEFine: REMark ~~~~~
```

avoid having to undo all that follows!

Go into display F3-D, and set the upper margin, page length, and bottom margin to zero, then set the header F3-H and footer F3-F to "none". Now go to the

reduce the tedium to a single altkeypress by inserting the following into your boot "ALTKEY'p',CHR\$(220)&"<greek mu>", and then all you have to do is to hold down the Alt key and keep pressing p until you

```

10 nm$= "Exp_Paras"
15 REMark (pd) hjc 1993.01.01 ver 1.1
20 REMark ~~~~~Main
25 Setup: Process: Finish: STOP
30 REMark ~~~~~
100 DEFINE PROCEDURE Setup
105 LOCAL L,lp: c%=3 : REMark Two unused channels
110 REPEAT lp
115 CLS: INPUT "Device and file name:" \,n$
120 L=LEN(n$)
125 IF n$(L-3 TO L)="_lis": n$=n$( 1 TO L-4)
130 IF LEN(n$)<13: EXIT lp
135 BEEP 25,3000: END REPEAT lp
140 OPEN_IN#c%,n$&"_lis": OPEN_NEW#c%+1,n$&"_haf"
145 eol$=CHR$(13)&CHR$(10): REMark MSDOS form
150 END DEFINE : REMark ~~~~~
200 DEFINE PROCEDURE Process
205 LOCAL a$,i,L,lp
210 REPEAT lp
215 IF EOF(#c%): EXIT lp
220 INPUT#c%,a$: IF a$="" : a$=" "
225 IF a$(1)=" "
230 L=LEN(a$)
235 FOR i=1 TO L: IF a$(i)<>" " :EXIT i
240 IF i<L: PRINT#c%+1,eol$&a$;
245 ELSE PRINT#c%+1," "&a$;
250 END IF : END REPEAT lp: END DEFINE :REMARK ~~~
300 DEFINE PROCEDURE Finish
305 PRINT#c%+1,eol$;
310 CLOSE#c%:CLOSE#c%+1
315 BEEP 5000,50: PRINT "DONE"
320 PRINT "New file name is ";n$&"_haf"
325 END DEFINE : REMark ~~~~~

```

get to the end.

Now you are ready to export your document. With the special printer.dat file in the right place (remember, you can copy it from within Quill using the "backup" facility <F3, F3, F, B, etc.>) "print" your document to a file <F3-P-Enter-Enter-file name>. This will give you a file with the text in the required format with the _lis extension tacked onto your name.

The method of using a SuperBasic utility requires the same "de-formatting" of the above, except that the program assumes that any line starting with a space is a new paragraph, and omits any blank lines. (This could cause problems, as paragraphs are best separated by a blank line. However, Howard's samples worked. Ed.) It also requires a special printer.dat file, similar to the above except that the end of line code and postamble should be the QL's standard, LF only, and the translate isn't necessary. You must first generate your _lis file using this printer.dat driver, and then leave Quill for SuperBasic and run the _lis file through listing six. Since it deals with the file a whole line at a time it is quite quick. The output will have the extension _HAF.

Public Domain

The programs listed in these articles are public domain: they may be freely copied, and distributed as long as no charge is made beyond a small copying charge. Copies will be available from the QubbeSoft public domain library for anyone who considers their own typing skills inadequate. Some additional, related material may also be available as part of the package as well as a collection of my public domain programs, including Ftidy, which was formerly available from the Microdive Exchange.

PSION Solutions

QUANTA

Independent QL/Thor Users Group

Worldwide Membership is by subscription only, and offers the following benefits:

- Monthly Newsletter – up to 40 pages
- Massive Software Library – mostly FREE!
- Free Helpline and Workshops
- Regional Sub-Groups. One near you?
- Advice on Software & Hardware problems
- Discounts from most major suppliers
- Subscription just £14 for UK members
- Overseas subscription £17

Barclaycard: Visa: Access: Mastercard

* Now in our NINTH successful year *

Further details from the Membership Secretary



Bill Newell
213 Manor Road
Benfleet
Essex. SS7 4JD
Tel (0268) 754 407



IF AN ADVERT IS WRONG, WHO PUTS IT RIGHT?

We do.

The Advertising Standards Authority ensures advertisements meet with the strict Code of Advertising Practice. So if you question an advertiser, they have to answer to us.

To find out more about the ASA, please write to Advertising Standards Authority, Dept. X, Brook House, Torrington Place, London WC1E 7HN.



This space is donated in the interests of high standards in advertisements.

Beginners' Machine Code

In part 3, Alan Bridewell starts moving memory about.

In the first part of this series, we just about managed to get something on the screen. In part two, we improved slightly with a pretty pattern, but not something of any great use. In this part I hope you will see a QL (that is, a Quantum Leap) in what we actually achieve.

First, we shall look closely at how the assembler uses labels, and show how this leads to the need for another type of instruction, the LEA instruction. You should recall that labels are what we put into the left hand column of an assembler listing. They represent the address of the code following the label in the program. An important (but not the only) reason for labels is that programs can be loaded anywhere in memory, so that at the time we write the program, we do not know what actual address the label refers to. The use of a label ensures that the assembler will generate machine code which will always refer to the correct address regardless of where the program is loaded. How it does this need not concern us.

Label syntax

In order to use labels correctly, we need to learn the syntax rules about the way we refer to labels in MOVE instructions in assembler code. Suppose we have a label called 'CA_GTLIN'. These rules are:

1. If we refer simply to the label, eg CA_GTLIN, then we use the contents of that address as a further address, and we deal with the contents of that new address.

2. If we put a hash (#) sign in front of the label, eg #CA_GTLIN,

then we treat the contents of that address as a number.

To make this clear, we shall look at the effect of each type of instruction. Suppose the label CA_GTLIN refers to an address which contains the number \$118. (This, by the way, is not one I made up. It has a special significance to Qdos, and your assembler may automatically accept the label CA_GTLIN as meaning the number \$118, even if you do not actually say so in your program. More about that later.)

The instruction:

```
MOVE.W #CA_GTLIN,D1
```

means "make the contents of the data register D1 into \$118 and discard what was already there".

The instruction:

```
MOVE.W CA_GTLIN,A2
```

means "take the contents of the address \$118, and copy it into the address register A2, discarding what was already there". If I actually did this on my usual QL, which is a JM, I would

get the number \$5C4C in the A2 register. On other QLs, Thors, Minervas or emulators you may get some other number. (I haven't tested this, so I can't be sure.) The point is, this is the

```

; *****
;                               SAVE AND RESTORE THE SCREEN
; *****
;
; SAVE SCREEN TO BUFFER
;
;                               MOVE.W    #$8000,D0          ; LOOP COUNTER
;                               MOVEA.L   #$20000,A0         ; SCREEN ADDRESS IN A0
;                               LEA.L     BUFFER,A1          ; BUFFER ADDRESS IN A1
;                               BRA        LOOP              ; SAVE SCREEN
;
; RESTORE SCREEN
;                               MOVE.W    #$8000,D0          ; LOOP COUNTER
;                               MOVEA.L   #$20000,A1         ; SCREEN ADDRESS IN A1
;                               LEA.L     BUFFER,A0          ; BUFFER ADDRESS IN A0
;                               MOVE.B    (A0)+,(A1)+        ; MOVE CONTENTS OF (A0)
;                               ; TO (A1), THEN
;                               ; INCREMENT BOTH
;                               DBRA      DO,LOOP            ; IF DO > 0, THEN DECREMENT DO AND
;                               ; BRANCH TO 'LOOP'
;                               ; IF DO = 0, THEN LEAVE THE LOOP
;                               ; AND CONTINUE THE PROGRAM
;                               MOVEQ     #$0,D0             ; NO ERROR RETURN
;                               RTS
;
; .BUFFER

```

```

100 z=RESPR(32810)
110 LBYTES f1p1_LISTING1_code,z
120 OPEN#3,scr_512x256a0x0
130 CLS#3
140 FOR n = 1E-2 TO 1 STEP 1E-2
150 CIRCLE#3,100*SIN(n),100*COS(n),100*n
160 END FOR n
170 sstore
180 PRINT#3, "This screen has been stored"
190 PRINT#3, "Now press a key to clear the screen"
200 PAUSE -1
210 CLS#3
220 PRINT#3, "Now press a key to restore screen"
230 PAUSE -1
240 srestore
250 CLOSE#3
260 PAUSE -1
270 DEFINE PROCEDURE sstore
280 CALL z
290 END DEFINE sstore
300 DEFINE PROCEDURE srestore
310 CALL z+18
320 END DEFINE srestore

```

address in rom for the sub-routine used to take long word parameters from SuperBasic procedures and functions. A different rom may have that routine at a different address, but that address can always be found at address \$118 in all rom versions. It's the use of numbers like this (called vectors) which make programs that use sub-routines in the rom compatible with different versions of the QL, but that is another story.

Load address

An important problem is that neither of these MOVE instructions will enable us to move the actual address of the label CA_GTLIN into a register for us to use. To do this, we need another type of instruction, LEA, which stands for 'Load Effective Address'.

The instruction:
LEAL CA_GTLIN,A2

means "make the contents of the A2 register into the address of the label CA_GTLIN". Note, this is NOT \$118, which is the actual contents of the address CA_GTLIN, and it is NOT \$5C4C, which is the contents of the address \$118 (on the JM QL, anyway).

An important use of a label is to refer to the address of the start of a block of memory which can be used temporarily to store data generated while the program is running. Such a block is usually referred to as a buffer. The most convenient place to put this is often immediately after the actual program, and we can do this by putting the label BUFFER (or any other suitable name) at the end of the listing. We can then use the program to store our generated data at the addresses starting at BUFFER. The only thing we must watch is to remember when we reserve memory for our program (usually using the RESPR command from SuperBasic), to make sure it is big enough to include both the program and the buffer we need. Otherwise we might find our QL has decided to use the buffer space for something else, with probable disastrous results.

Block copy

To fill such a buffer with data, we can use a "post incrementing" instruction to

advance from one address to the next, in much the same way as we did in the previous part of this series. However, at that time we only did this to the destination address, which meant we copied the same data into all the bytes. But this type of instruction can be used to increment both the source and destination, and so copy a whole block of memory into

another block. Let us see how this works.

Suppose address register A0 contains \$20000, which, you may recall, is the first address of the screen ram. Next, suppose register A1 contains the first address of our buffer. The instruction

MOVE.B (A0)+,(A1)+

means "take the contents of the address \$20000 and copy it into the first address of the buffer, and then increment both

addresses". The next time we use the same command it will mean "take the contents of address \$20001 and copy it into the second address of the buffer, then increment both addresses". Clearly, by putting this instruction into a suitable loop, we can copy the entire screen ram into our buffer as a temporary store. The DBRA instruction used last time will do fine for this if we make register D0 contain \$8000 to act as a loop counter.

But we can do more than this.

LISTING 3

```
100 REMark Sinclair QL World HEX LOADER v 3
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
130 CLS: RESTORE :READ space:start=RESPR(space)
140 PRINT "Loading Hex...":HEX_LOAD start
150 INPUT "Save to file...";f$
160 SBYTES f$,start,byte:STOP
170 :
180 DEFine FuNction DECIMAL(x)
190 RETURN CODE(h$(x))-48-7*(h$(x)>"9")
200 END DEFine DECIMAL
210 :
220 DEFine PROCedure HEX_LOAD(start)
230 byte=0:checksum=0
240 REPEAT load_hex_digits
250   READ h$
260   IF h$="*":EXIT load_hex_digits
270   IF LEN(h$) MOD 2
280     PRINT "Odd number of hex digits in: ";h$
290     STOP
300   END IF
310   FOR b=1 TO LEN(h$) STEP 2
320     hb=DECIMAL(b):lb=DECIMAL(b+1)
330     IF hb<0 OR hb>15 OR lb<0 OR lb>15
340       PRINT "Illegal hex digit in: ";h$:STOP
350     END IF
360     POKE start+byte,16*hb+lb
370     checksum=checksum+16*hb+lb
380     byte=byte+1
390   END FOR b
400 END REPEAT load_hex_digits
405 PRINT checksum:STOP
410 READ check
420 IF check<>checksum
430   PRINT "Checksum incorrect. Recheck data. ":STOP
440 END IF
450 PRINT "Checksum correct. Data entered at: ";start
460 END DEFine HEX_LOAD
470 :
480 REMark Space requirements for the machine code
490 DATA 42
500 :
510 DATA "303CB000": REMark      MOVE.W      $$80000,D0
520 DATA "207C00020000": REMark  MOVEA.L      $$20000,A0
530 DATA "43FA001E": REMark      LEA.L        BUFFER,A1
540 DATA "60000010": REMark      BRA          LOOP
550 DATA "303CB000": REMark      MOVE.W      $$80000,D0
560 DATA "227C00020000": REMark  MOVEA.L      $$20000,A1
570 DATA "41FA000C": REMark      LEA.L        BUFFER,A0
580 DATA "12D8": REMark          .LOOP      MOVE.B      (A0)+,(A1)+
590 DATA "51CBFFFC": REMark      DBRA        D0,LOOP
600 DATA "7000": REMark          MOVEQ       #$0,D0
610 DATA "4E75": REMark          RTS
620 DATA "*, "2905"
```

Suppose at some later time our screen has become altered, either by clearing it, or otherwise corrupting it. If we now make register A0 contain the first address of the buffer, and register A1 contain the first address of the screen ram, we can use the same loop to copy the contents of the buffer back to the screen ram, thereby restoring the screen to its former glory. **Listing one** shows how this is done. Much of it is very similar to the listing in part two, so if you understood that, then this should give you few problems.

The first line puts \$8000 (the size of the screen ram) into register D0 to act as a loop counter. The second line puts \$20000 (the start of the screen ram) into register A0. The third line uses the new LEA instruction to put the address of the buffer into register A1. Note that the label BUFFER is at the end of the listing, so the first address of the buffer will come after the last address of the program.

Branch always

Next comes another new instruction BRA which simply stands for "branch always". It means exactly that - whatever happened before, the next step is to branch to the address given by the label in the operand, LOOP in this case. This loop is the one which copies the data from screen to buffer, and should need no further explanation. What does need explaining is why the loop is not the very next bit of code, so removing the need for the BRA instruction.

The reason is that the loop is to be used not only to copy the screen to buffer but also to copy the buffer back to the screen. This requires the registers to be set up in a slightly different way, and so the next three instructions (5th, 6th and 7th) give the alternative starting point used for restoring the screen. The three instructions are exactly the same as the first three except that registers A0 and A0 have been swapped over.

When the program is assembled, we need a SuperBasic program to CALL the code to see how it works.

Listing two is a simple example. It loads the code, produces a distinctive pattern on the screen, saves it in the buffer,

clears the screen, and finally restores the pattern from the buffer. Note the large parameter in the RESPR command. This is because we need to reserve enough ram for the code (which is only a few bytes long), plus the whole screen ram, which is 32 kilobytes long!

Disassembly

There is an extra complication here which did not occur in the previous SuperBasic programs. We have two starting points in our code, one for copying the screen to the buffer, and one for copying the buffer to the screen. Copying the screen to the buffer is no problem because we start from the first address of our machine code, which is, of course, the address used for the LBYTES

command which was obtained by the RESPR command. But how do we know that if the program loads at address z, then the address to CALL to copy the buffer back to the screen is (z + 18)?

The only way to do this is to use a disassembler, which is a program to do the opposite to the assembler - it converts machine code into something like the original assembler listing, together with the memory addresses.

MOVE.W #\$8000,D0

instruction was \$12 from the start of the program, and, converting that into normal decimal notation, gives us 18.

If you do not have a disassembler, then you really should get one if you seriously want to do machine code programming. Quite apart from the use I have made of it here, it is a quite invaluable tool for debugging your program to find the simple little mistakes every programmer makes all the time.

However, provided you have copied Listing one correctly, and successfully assembled it, the CALL address I have used in Listing two should be correct.

There is one possible problem with some assemblers, though. There is an alternative version of the BRA instruction, BRAS. This stands for "short branch always", and can be used when the address we are branching to is close by, which it is in our program. This alternative is actually preferable, because it results in a shorter program which runs faster (though not noticeably in this program). I have not used it because it is an extra complication I would rather leave to later.

The problem is that some assemblers will automatically use BRAS if possible, even if you actually used BRA in the program, in order to make the program better than the one you actually wrote. If this happens, you will need to alter line 310 in Listing two to

CALL z + 14

As in the previous

articles, I have assumed that the machine code is in a file LISTING1_code in flp1. You may need to alter this to suit your circumstances. Also, for benefit of those without an assembler, I have Marcus and Simon's hex loader set up to generate the machine code.

Once you have got the machine code to work, there are one or two little experiments you can try.

Word at a time

First, we don't have to transfer data a byte at a time. We can do it a word at a time, or a long word at a time. To transfer a word at a time simply requires the LOOP line to be altered to

MOVE.W (A0)+(A1)+

Because the data is now moving two bytes at a time, it only requires half as many steps, so the lines setting the loop counter must be altered to

MOVEQ #\$4000,D0

This should make the code run faster, although it runs so fast already, you will be hard put to notice the difference, unless you repeat it over and over again in rapid succession. If you have understood this alteration, you should be able to work out for yourself how to alter the code to transfer the data a long word at a time and make it faster still.

Another interesting variation to the program is this: instead of copying the whole screen to a buffer, we could copy one part of the screen to another part of the screen, provided the parts of the screen consist of consecutive screen bytes. For example, to copy the top half of the screen to the bottom half and vice versa, we need these alterations.

Instead of the LEA instructions for loading the buffer address, we can use

MOVEAL #\$24000,A0

which loads the address half way down the screen. Since we are only moving half a screen at a time the loop counter line must be changed to

MOVEQ #\$4000,D0

Since we are not using a buffer, we can do away with the BUFFER label in the code, and when we load the machine code we can make do with 32K less memory.

If you get this to work, you should then be able to write versions to transfer parts of the screen either a word or a long word at a time.

Any part

Clearly, with a few simple changes, this routine can be used to copy any part of the memory (ram or rom) to any other part of the ram. (However, we must be careful about where in ram we put it, or Qdos may object, and crash.) I said at the outset that I would make all the examples in this series do something on the screen, to make it obvious what had been done, which is why I have used the routine in this particular way.

I hope you will agree that what we have here is a simple routine that really can be used to improve our SuperBasic programs, the promised Quantum Leap.

SuperBasic In Action

Simon Goodwin explains how most printers can be persuaded to print QL screens in colour.

This SuperBasic project uses the powerful RECOL command to separate the colours on a QL screen, so that it can be printed in successive passes with different coloured ribbons or ink cartridges. The technique works with most printers that can only print one colour at a time, including cheap matrix printers.

The only types not supported are thermal printers like the Serial 8056, which cannot be persuaded to burn their paper to any colour other than the default, usually black, and daisywheels, which do not support graphics printing.

You do not need a special colour driver, like Psion's JX80 routine for colour Epsoms. You can use the normal GPRINT.PRT from your Easel cartridge, or commercial alternatives like TASCOPY or ULTRAPRINT, which are faster and can print the whole area of the screen.

Ink-change

The only catch in this method, compared with true colour printing, is that you need to reposition the paper and change the ink for each component colour. It's a reasonable economy if most of your printing is monochrome and you only need colour images occasionally.

I can't afford a full-colour printer, but I can generate colour prints with the Deskjet 500, using a succession of different coloured ink cartridges. These are easier and cleaner to change than coloured printing ribbons, and the Deskjet's sheet feeder means that successive colours are correctly aligned.

The examples use the HPDUMP command introduced in DIY Toolkit last month, but could just as well use GPRINT.PRT. To run this from SuperBasic you need to load it into memory allocated with RESPR or ALCHP, and CALL the start each time you want to send a dump to the printer.

These commands allocate memory, load the GPRINT file from floppy disk 1, and print the current screen:

```
X=RESPR(640)
LBYTES FLP1_GPRINT.PRT,X
CALL X
```

The size of the GPRINT file may vary depending on your printer and version of the Psion software, but 640 bytes is enough for all the versions I have encountered.

Toolkit functions like ALCHP, RESERVE and ALLOCATION let you deallocate the memory when the printing is done, whereas RESPR memory remains allocated till you reset the computer. Use RECHP X to deallocate ALCHP space, DISCARD X for DIY Toolkit, and DEALLOCATE X for DIY Toolkit.

GPRINT.PRT scans are 480 pixels wide, wide enough for EASEL displays and TV screens but 32 pixels less than the full Mode4 screen. Later Epson dump routines like SIDEWINDER and TASCOPY avoid this limitation of 80-column printers by printing the screen down rather than across the paper.

Psion's GPRINT.PRT was only designed for nine-pin printers, but Dilwyn Jones's *Trans 24* utility can translate its output to suit more modern 24 pin models.

Alternatively Dilwyn's *Image Processor* can drive most modern printers directly.

Saving Screens

Psion's *Easel* can save screens to a file instead of the printer, adding the extension .PIC to the device and file name you supply. If you have access to SuperBasic you can save the QL screen image at any time with this command:

```
SAVE
FLP1_MY_SCREEN,131072,32
768
```

The numbers give the start and size of the screen memory area. Use 163840 instead of 131072 if you want to save Minerva's second screen.

If SuperBasic is busy, build these commands into a compiled task triggered by a key-press, or invest in Dilwyn Jones's multi-tasking *Screen Snatcher*, designed for just this purpose.

Mode4 colour

To print Mode4 screens in full colour on white paper, you need red and green ink cartridges as well as the usual black one. Start with the red cartridge, and set a full-screen window with WINDOW 512,256,0,0. Load the Mode4 screen, with LBYTES, then type:

```
RECOL 0,0,7,7,0,0,0,0 :
HPDUMP
```

The RECOL replaces all the red in the screen with white, and then prints that, with brightness inverted so that only the red part of the picture is inked. By default HPDUMP generates an 'reversed' monochrome image, with black on white print for white on black displays. Replace 0s with 7s, and vice versa, in the RECOL

command if your dump routine does not reverse the colours. The next pass uses the Green ink cartridge. Move the piece of paper from the top to the bottom tray, so it can be overprinted with green ink. Re-load the original screen before you issue the RECOL command, as the green information will have been eliminated by the red print pass. These commands add the Green part of the picture:

```
RECOL 0,0,0,0,7,7,0,0 :
HPDUMP
```

The final stage, in Mode4, is to add the black:

```
RECOL 7,7,0,0,0,0,0,0 :
HPDUMP
```

Coloured ink cartridges are available from many suppliers, and it is cheap - if a little fiddly - to re-fill used cartridges with fresh ink. You can usually find suppliers at an All Formats Fair; I obtain my supplies from Mechanical Services, 11 Elisabeth Road, Sutton Coldfield, West Midlands B73 5AR. They can also supply special inkjet paper which copes well with overprinting, and coloured ribbons for matrix printers - call 021 354 3299 to check if your printer is supported.

Mode8 colour

Mode8 pictures can be processed in a similar way, but there is no need to use seven different ink colours. As long as you print the lightest colours first, you can generate all the QL colours by successive overprinting in yellow, cyan and magenta.

You get better results if black parts of the picture are printed with a black cartridge, rather than a murky mixture of the three secondary colours. This is the technique used to print QL World's coloured covers.

In theory the Mode4 dump could mix red and green to get black, but this is hardly worthwhile as it spoils the result, and most printers come with a black cartridge or ribbon as standard.

Conversion from the QL screen's RGB (Red Green Blue) colours to printer-friendly CYMK (Cyan, Yellow, Magenta and black) involves a similar sequence of RECOL statements and dump commands.

Once again we recolour required parts of the picture to show where the ink should go,

but this is complicated because Mode8 display memory uses a different internal code for white, and the separations must be converted to their Mode4 equivalent before HPDUMP is called.

One in every four bits of a Mode8 screen is reserved for 'flash' information, so white areas use 12 set bits in Mode8, rather than all 16 in Mode4. If we print this with a Mode4 dump routine like HPDUMP or TASCOPY_MON, there will be stripes across areas that should be all the same colour.

This is no problem with a Mode8 routine like TASCOPY_TV, which ignores the flash bits, but it needs attention if you want to use GPRINT_PRT or HPDUMP.

The trick is to recolour the required part of each separation into magenta on a Mode8 screen, then save and re-load the image as Mode4, where it will appear as red, rather like a Qdos cursor does when you change mode. Once in Mode4 we can convert the red to white, either with RECOL or faster with the SET_GREEN -1 command from *DIY Toolkit's Windows* volume W.

The listing takes a 32K screen file and converts it into four separate screens, sent to the

printer in sequence. It uses a temporary file to move the image from Mode8 to Mode4, and loads the original screen four times, so delays are minimised if you keep all the files on ram disk. Edit the name in A\$ to suit your original file and change the temporary name in B\$ if you like.

The lightest colour, yellow, is printed first and black is added last, as this gives the best resolution. The program waits for you to press Enter to signal that you have changed the ink before each pass.

When coloured inks are printed on top of one another only the common component is reflected, so green uses cyan on top of yellow, and blue is made from cyan and magenta. If magenta and yellow inks are mixed you get red. The other Mode8 colours are white - the paper colour - black, from a separate ink cartridge, and the secondary ink colours themselves.

Other models

I have not tried Mode8 colour printing on a dot-matrix printer but my friend Andy Wright reports success with four coloured ribbons and an old Epson RX-80

printer. Ribbon changes can get messy, and you need some way to ensure accurate alignment. Use a sheet feeder, or an alignment mark on the printer body, but make sure that the paper is tensioned the same way before each pass.

Laser printers can use coloured toner, equivalent to coloured ink, but such toner cartridges are relatively expensive. I don't know anyone who uses multiple laser passes to print in full colour, but the principle should be the same as for other types of printer. Whatever you use, let us know how you get on.

```
100 REMark MODE 8 COLOUR SEPARATIONS
110 REMark V2, for any mono QL screen-dump
120 REMark Copyright 1993 Simon N Goodwin
130 :
140 a$="raml_Arbitrary_Screen"
150 b$="raml_Temp"
160 screen=131072 :REMark 163840 for Minerva #2
170 fullsize=32768
180 MODE 8
190 OPEN #3,"SCR_512x256a0x0"
200 INPUT #0;"Press ENTER for YELLOW pass ";c$
210 SEPARATE 0,0,3,0,3,0,3,0 :REMark Yellow
220 INPUT #0;"Press ENTER for CYAN pass ";c$
230 SEPARATE 0,3,0,0,3,3,0,0 :REMark Cyan
240 INPUT #0;"Press ENTER for MAGENTA pass ";c$
250 SEPARATE 0,3,3,3,0,0,0,0 :REMark Magenta
260 INPUT #0;"Press ENTER for BLACK pass ";c$
270 SEPARATE 3,0,0,0,0,0,0,0 :REMark Black
280 CLS #3
290 CLOSE #3
300 :
310 DEFine PROCEDURE SEPARATE(a,b,c,d,e,f,g,h)
320 MODE 8
330 LBYTES a$,screen
340 RECOL #3,a,b,c,d,e,f,g,h
350 SBYTES b$,screen,fullsize
360 MODE 4
370 LBYTES b$,screen
380 DELETE b$
390 REMark SET_GREEN #3,-1 or
400 RECOL #3,0,0,7,0,0,0,0
410 HPDUMP :REMark or TASCOPY_MON etc.
420 END DEFine SEPARATE
430 :
440 DEFine PROCEDURE S
450 SAVE SEPARATOR_BAS
460 END DEFine S
```

ARCHIVE YOUR QL COLLECTION

Now you can keep your Sinclair QL World magazines safe and clean. No more dog-eared covers or missing copies . . . You can protect your magazines in this high quality, specially-created binder. This Sinclair QL World binder will comfortably hold a complete year's issues of your favourite Sinclair magazine. It is a high

quality product, British-made and comes with full binding instructions. It is manufactured in a rich, deep blue with genuine gold blocked lettering. Enhance your Sinclair QL World Magazine collection now for only £6.95 (inc.P&P!) *Send for one today!* The QL World binders also make an ideal gift for other Sinclair users too!

TO: QL WORLD, THE BLUE BARN, WOOTTON, WOODSTOCK, OXON. OX7 1HA

Please send me ☐ QL World binders - I enclose £6.95 for each binder including VAT: postage & packing.

Readers outside the UK and Eire please add £1.95 for surface overseas mail.

Please make cheques payable to ARCWID LTD.

ACCESS ☐ VISA ☐ account:

Expiry date:.....

Name

Address

Postcode

Tel No.

SINCLAIR **QL** WORLD

JUNE
45

DILWYN JONES COMPUTING

41 BRO EMRYS, TAL-Y-BONT, BANGOR,
GWYNEDD, LL57 3YT, GREAT BRITAIN

TELEPHONE: (0248) 354023

QL SOFTWARE

A SELECTION FROM OUR RANGE OF NEARLY 100 PRODUCTS FOR THE QL, NO
ROOM TO ADVERTISE THEM ALL HERE, SO PLEASE ASK FOR OUR
CATALOGUE (PHONE FOR A COPY OR SEND A LETTER WITH YOUR ADDRESS).

EASYPTR III part £40.50

Simplified pointer environment programming. Part I consists of sprite editor, menu editor and superbasic extensions to use in your own programs. Applications created using Easyptr III can be compiled with QLiberator. Requires expanded memory, available on disk only.

EASYPTR III part 2 £20.00

Consists of appendix manager and enhanced toolkit for control of menus etc in your programs. EASYPTR III part 3 £20.00

Consists of Easysource and C library routines etc.

QLIBERATOR £50.00

Superb superbasic compiler, compiles virtually all of basic plus most toolkit commands, etc. Produce faster multitasking code from your basic programs. Compile resident extensions, use overlays etc. with the latest V3.36. Can be mouse controlled. Expanded memory required.

BUDGET QLIBERATOR £25.00

Excellent value, virtually all of superbasic but without some of the additional facilities of the full version. Not mouse controlled. Works on unexpanded QL too.

DJTOOLKIT £10.00

Compact toolkit of BASIC extensions, ideal for use with ALiberator. Really useful programming commands, can be distributed with your compiled programs if you wish. At this price, a bargain! Suitable for unexpanded QL.

LINEDESIGN £100.00

Vector drawing package, uses outline fonts and clipart, move and resize text and graphics without loss of quality. Ideal for making posters, etc. Supplied with huge range of fonts and clipart on TEN disks! The more memory your system has, the better! Available on disk only, can be mouse controlled (including SERmouse).

DATA DESIGN 3 £60.00

Superb, fast pointer driven database with free form field structures, with the option of disk based for large files if required, or smaller files can be kept in memory for speed. You do not have to be able to program this version but if you add the API package, it can be programmed from basic, C, or assembler. Expanded memory required, disk only.

API for Data Design £20.00

QPAC2 £39.95

Tony Tebby's superb pointer environment package. In addition to the pointer environment files themselves, this includes tutorials, extensive manual, files menu, channels and jobs menus, easy switching between jobs, hotkeys, etc. Mouse or keyboard controlled, a good introduction to pointer environment, 256k ram minimum, disk only

QPAC1 £19.95

Ideal companion to QPAC2, consists of small accessory programs such as calculator, calendar, clocks, alarm clocks, typewriter, etc. All can be mouse controlled. Pointer environment files included. Can be used with or without QPAC2. Expanded memory required, disk only.

QTYPE2 £29.95

Tony Tebby's spelling checker program. Check spelling as you type OR check existing files retrospectively. User interface allows you to write programs which use the dictionary facilities. English, French and German dictionaries included!

DISA £29.00

Interactive pointer driven machine code disassembler. 256k ram min. Disk only.

MEGATOOLKIT £25.00

EPROM VERSION £40.00
Large toolkit with over 200 BASIC extensions, suitable for use with QLiberator or Turbo. Many examples supplied, extensive manual. Suitable for unexpanded QL.

DISCOVER £20.00

The painless way to move files from QL to PC and vice versa. As simple as copying files between two disks. 256k ram min., disk only.

MULTI DISCOVER £30.00

In addition to Discover facilities, also contains CPM, Unix CP10, BBC micro and now Spectrum and SAM Coupe file transfer capability. 256k min. ram, disk only.

TEXTIDY £15.00

Assists Discover with conversion of text files by stripping out control codes etc. 256k ram min.

CONVERT-PCX £10.00

Used with Discover, allows transfer of bit mapped PC clipart graphics in PCX format (a common PC file format) to QL screens or Page Designer pages. 256k ram, disk only.

QL-PC-FILESERVER £24.50

Link a PC and a QL via a serial port cable and use this software to enable the two to communicate - the QL can save its files on a PC's disk systems and print to the PC's serial port using normal basic commands like COPY. Works on unexpanded QL.

BANTER £25.00

Simple to use banner maker which uses outline fonts for good quality large texts. Prints sideways across up to 4 sheets of paper. Simple to use, menu driven, on screen preview before printing, etc. Suits most Epson compatible printers.

IMAGE PROCESSOR 2 £15.00

Easy to use graphics system, featuring usual graphic facilities, pixel zoom editing, image enhancement, mode conversion etc. 512k, disk only.

SCREEN COMPRESSION £10.00

Reduce the amount of storage required by graphics on disk or microdrive - supports several QL formats. 256k, disk only.

SCREEN DAZZLER £15.00

Unlike the usual screen savers, which simply turn off the display when the keyboard is not used for a while, this one can activate various graphical displays to provide an attractive means of preventing screen burn-in, more like the screen savers on other computers. If you have a compiler, you can even write your own savers by following the instructions in the manual. Pointer environment compatible.

SCANNED CLIPART 1 £10.00

NEW! A disk full of compressed scanned pictures (decompression program supplied of course) which can be used in most QL programs (DTP, graphics, etc). Assorted collection, containing many pictures you may not find in other collections. Large number of pictures, a bargain at this price. 128k, disk only.

PRINTERMASTER £20.00

Select printer control codes quickly and simply from a menu to set fonts, page lengths, etc. before printing from programs like Quill, etc. 128k, disk/mdv

SERMOUSE £40.00

Albin Hessler's serial mouse driver system for the QL is now available from DJC complete with a QL style matching black mouse with 9 pin serial connector and UK style port adaptor lead. Version 3 driver software. Can now work with The Painter too. NB POSTAGE CHARGES BELOW

MAGAZINES EX-CGH SERVICES

Ask for a price list of back issues of QL Technical Review, QL Leisure Review and QL Adventurer's Forum (all available at time of writing).

SQUIDGY ROUND THE WORLD £12.50

An arcade game, ideal for the young at heart! 128k

5-GAMES PACK £12.50

5 'thinking' games in one bargain pack, 128k

SUPPLIES

FLOPPY DISKS £0.40

DSHD DISKS £0.70

MICRODRIVES £2.50

DISK LABELS £2.00

On printer roll £2.50

ADDRESS LABELS £2.00

MDV LABELS £2.00

MOUSE MATS £2.50

Disk box dividers £3.00

in stock once more!

TERMS: Discounts - buy 2 programs, claim 5% off each, buy 3 or more, claim 10% off each program. Offer applies to software only. **POSTAGE** - Software is sent post free to UK addresses. Overseas please add £1.00 per program for postage (maximum £3.00). Floppy disks and serial mouse - add postage of £2.50. Labels/mouse mats - add postage of £0.50 per item if only buying these. **PAYMENT** - In UK currency (pounds sterling) only please. Payment by cheque, Eurocheque, Postal Order, cash (send by registered post), or by credit card (Visa/Mastercard/Eurocard/Connect). In case of difficulty contact us first to arrange a payment method if none of these is possible for you. Please make cheques, etc payable to DILWYN JONES COMPUTING (not to any other name or abbreviation please, our bank prefers it that way!).